MIPS® BusBridge™ 3 Modules User's Manual

Document Number: MD00660
Revision 02.02
April 17, 2014

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The MIPS® BusBridge™ 3 Modules family provides high-performance OCP 2.1 interfaces between a MIPS core and other subsystems in an SOC.

This document is designed to serve as a high-level description of the MIPS BusBridge 3 Modules. This chapter provides an overview of the design and feature set of the various modules, and the ways they can be used.

This chapter is organized as follows:

1.1 Features

The MIPS BusBridge 3 Modules family includes the following blocks:

- OCP-to-AXI Bridge (OCP-AXI)
- AXI-to-OCP Bridge (AXI-OCP)

In addition, some products are delivered with an additional Reference Design for a dual AXI bridge (OCP-AXI2), which can be built with the OCP Splitter (OCP-SPL) and two instances of the OCP-AXI. Consult the MIPS softcore product datasheet to determine whether the OCP Splitter is included in the release.

1.1.1 OCP2AXI Bridge (OCP-AXI)

This module is a bridge between an OCP 2.1 master and an AMBA AXI slave. The main features of this bridge are:

- Connects to the master OCP port (v2.1) of the following MIPS® product families.
  - MIPS32® 24K®
  - MIPS32® 34K®
  - MIPS32® 74K™
  - MIPS32® 1004K™ Coherent Processing System
  - MIPS® SOC-it® L2 Cache Controller
  - MIPS32® P5600 core family

- AXI interface is compliant to AMBA AXI v1.0
- Support for 32-bit or 40-bit address, and 64-bit, 128-bit, or 256-bit data path
- No combinational paths from AXI to OCP flow control signals. This allows for full flexibility in closing SOC timing.
- Single-cycle latency on request path. No additional latency on the response path.
- Clocked with AXI system clock
    - AXI clock needs to be a supported synchronous divisor of OCP master's clock
- Fully synthesizable Verilog design

1.1.2 OCP Splitter (OCP-SPL)

The OCP Splitter is not a standard design. Consult the MIPS softcore product datasheet to determine whether the OCP Splitter is included in the release.

This module splits the slave OCP port into two master OCP ports based on a user-defined address decode. This enables an SOC designer to direct bus transactions from the MIPS32 core to two different subsystems. The main features of the OCP-SPL are:

- Connects to the master OCP port (v2.1) of the following MIPS product families:
    - MIPS32® 24K®
    - MIPS32® 34K®
    - MIPS32® 74K™
    - MIPS32® 1004K™ Coherent Processing System
    - MIPS® SOC-it® L2 Cache Controller
    - MIPS32® P5600
- User-configurable address decode
- Configurable port priority
- Supports OCP response flow control
- Combinational block with no additional latency
- Fully synthesizable design

1.1.3 AXI-to-OCP Bridge (AXI-OCP)

This module is a bridge between an AMBA AXI master and an OCP 2.1 compliant slave. It facilitates connection of an AMBA AXI based subsystem to the OCP 2.1 slave interfaces of the I/O Coherence Unit (IOCU) on the MIPS32

1004K Coherent Processing System (CPS) or the ScratchPad RAMs DMA interfaces on MIPS32 cores. The main features of the bridge are:

- Connects an AMBA AXI master interface to the OCP 2.1 slave interfaces of the ISPRAM, DSPRAM, and IOCU blocks on the following MIPS32 cores:
  - MIPS32® 24K®
  - MIPS32® 34K®
  - MIPS32® 74K™
  - MIPS32® 1004K™ Coherent Processing System (CPS)
  - MIPS32® P5600
- Support for 32-bit or 40-bit address and 64-bit, 128-bit, or 256-bit data path
- No combinational paths from OCP to AXI flow-control signals. This allows full flexibility in closing SOC timing.
- Single-cycle latency on request path. No additional latency on the response path.
- Clocked with AXI system clock
  - If the AXI-OCP bridge is connected to the IOCU of the MISP32® 1004K™ CPS, AXI clock has to be derived with a supported synchronous divisor of the CM clock. Please refer to the 1004K™ CPS Users Manual for details of the supported divisors.
  - If the AXI-OCP bridge is connected to DMA interface of the ScratchPad RAMs on MIPS32 cores, AXI clock has to be derived with the same supported synchronous clock divisor as on the main OCP port of the core. Please refer to the appropriate MIPS32 Core Integrator's Guide for more details.
- Fully synthesizable Verilog design

1.2 OCP-AXI and AXI-OCP Usage

This section describes the different use models for the OCP-AXI and the AXI-OCP bridges with the supported MIPS product families.

1.2.1 MIPS32® Core

In the most basic configuration, the OCP-AXI bridge connects to a MIPS32 core's system OCP interface directly, as shown in Figure 1.1. Note that the arrow directions in the figure are used to indicate master/slave interfaces and not data movement; all master interfaces have arrows pointing away from them. Note also that the AXI-OCP bridge is only needed if the MIPS32 core is configured with Scratchpad RAMs (ISPRAM/DSPRAM). The MIPS32 core can be one of the following:

- MIPS32® 24K® core family
- MIPS32® 34K® core family

- MIPS32® 74K™ core family
- MIPS32® 1004K Coherent Processing System
- MIPS32® P5600 core family

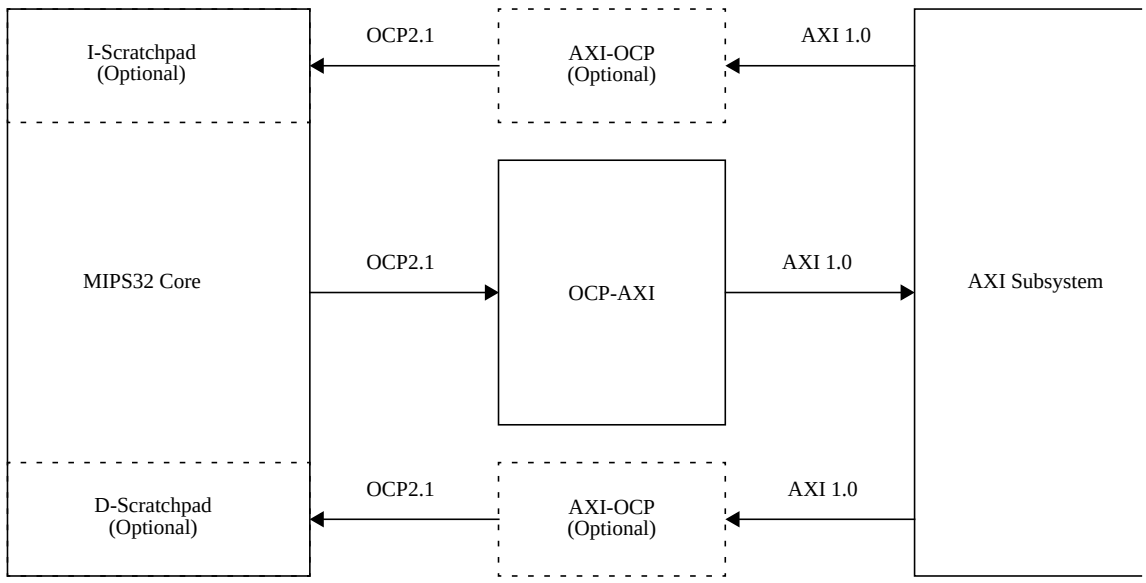


Figure 1.1 MIPS32® Core Connection

1.2.2 MIPS32 Core with SOC-it L2

In this type of system, where the MIPS32 core includes a SOC-it L2 Cache Controller, the OCP-AXI bridge connects directly to the SOC-it L2 cache controller's system OCP interface, as shown in Figure 1.2. The optional scratchpad connections are the same as those described in Section 1.2.1 "MIPS32® Core".

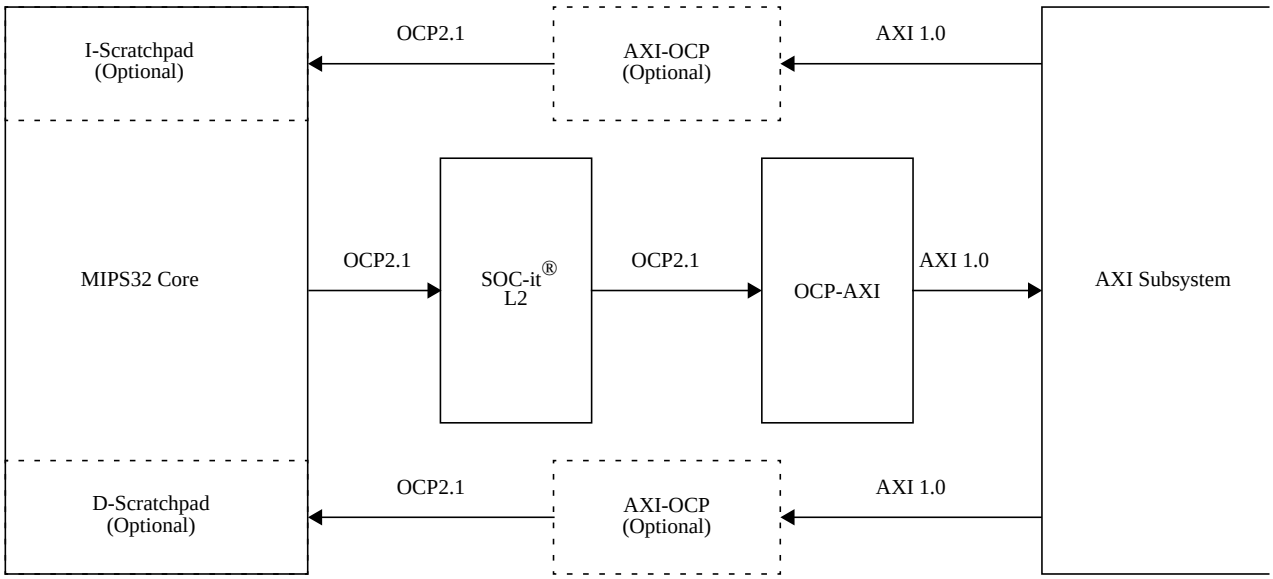


Figure 1.2 MIPS32® Core with SOC-it® L2 connection

1.2.3 1004K Coherent Processing System AXI Bridge Configuration

In a MIPS32 1004K Coherent Processing system (CPS), a number of AXI bridges may be required. To begin, each processor core may require bridges for optional scratchpad RAMs. In addition, the following bridges may be needed:

1. If the system is not configured with an optional SOC-it L2, then an OCP-AXI bridge is connected to the Coherence Manager block's system OCP interface. Otherwise, if a SOCit-L2 is present, then the OCP-AXI bridge connects to the SOC-it L2 cache controller's system OCP interface.

2. If an optional I/O Coherence Unit (IOCU) is included, then two AXI bridges may need to be added to that block (a master port and a slave port).

Note that for simplicity, Figure 1.3 shows the maximum configuration for a multi-core system, with all optional blocks shown as shaded.

1.3 OCP-SPL Usage

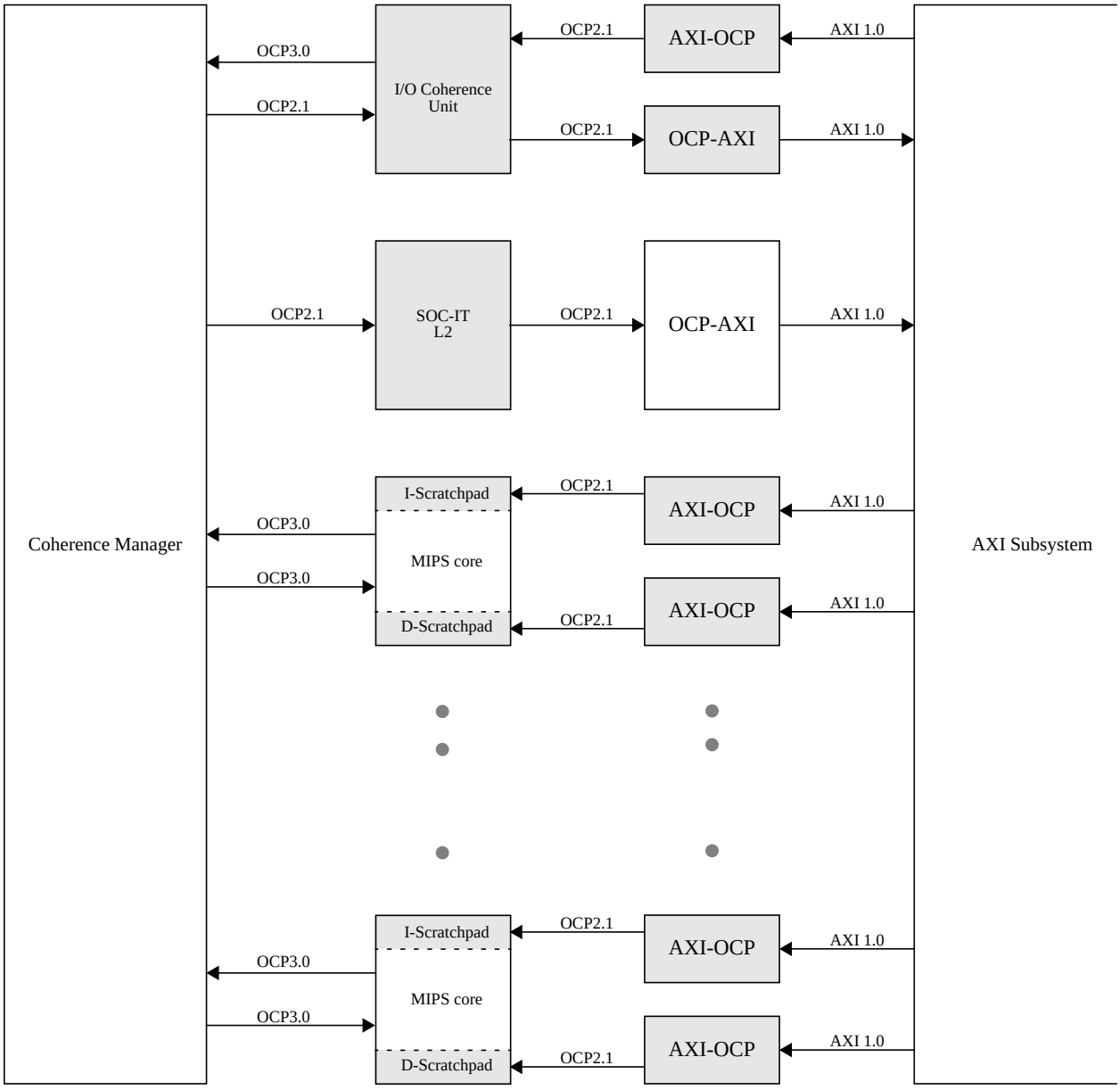


Figure 1.3 MIPS32® 1004K™ Coherent Processing System connection

This section describes the different use models for the OCP-SPL with supported MIPS products. The OCP-SPL can be used with the following MIPS IP blocks by connecting to the OCP master interface of the MIPS IP block.

- MIPS32® 24K® core family
- MIPS32® 34K® core family
- MIPS32® 74K™ core family
- MIPS32® 1004K™ Coherent Processing System
- MIPS® SOC-it® L2 Cache Controller
- MIPS32® P5600 core family

Chapter 2

Installation

This chapter describes the installation of the BusBridge™ 3 Modules package and its contents, and consists of the following sections:

- Section 2.1 “Tool Assumptions”
- Section 2.2 “Installing a Release”
- Section 2.3 “Directory Structure of the Deliverables”

2.1 Tool Assumptions

The deliverables associated with the MIPS® BusBridge 3 Modules assume the availability of a minimal set of UNIX-based scripting tools and front-end Electronic Design Automation (EDA) tools, as described in this section.

Use of the deliverables requires a x86 Linux operating system platform. Supported versions of the operating systems and tools are summarized in Section 2.1.3 “Supported Tool Versions”.

2.1.1 General Tool Requirements

The following scripting tools are assumed to be available at the implementor’s site:

- `csh`: csh/tcsh are used internally. All of the reference commands in this manual are given in C shell syntax and the various Makefiles depend on the use of csh. The deliverables have not been tested with bash, ksh, or any other shell program.
- `Perl`: Perl version 5 is required for some scripts included in various subdirectories of the release. Perl version 5 can be downloaded from http://www.perl.com.
- `make`: All Makefiles included in the release have been written to work with the GNU make tool. GNU make can be downloaded from the GNU web page at http://www.gnu.org.
- `Tcl/Tk`: The configuration tool assumes the availability of Tcl/Tk 8.0 or 8.1 for its graphical user interface. Tcl/Tk can be downloaded from the Tcl Developer Xchange web site at http://www.tcl.tk.
- C compiler: Compilation of MIPS-provided PLI routines requires a native C compiler. MIPS uses gcc, available at http://www.gnu.org. The deliverables have not been tested using other compilers.
- Binary tools: Compilation of MIPS-provided PLI routines requires the GNU linker and other GNU binary utilities. These are available at http://www.gnu.org.

2.1.2 Supported EDA Tools

The following is a list of front-end EDA tools directly supported with the MIPS BusBridge™ 3 Modules deliverables:

- Functional simulation: The RTL code is written in Verilog. The simulation environment includes support for the VCS simulator from Synopsys.
- Synthesis: The synthesis scripts support Synopsys DesignCompiler (DC).

There are no tool requirements for the back-end EDA tools that may be used to create a physical implementation of the MIPS BusBridge 3 Modules.

2.1.3 Supported Tool Versions

Table 2.1 shows the tool and platform versions with which the 24Kc core deliverables have been developed and tested internally at MIPS Technologies. The core deliverables should generally work with newer tool versions, but may not work with older ones. Synthesis and backend flow have been tested with the named versions. Newer versions have not been tested with the flows but should generally work.

Table 2.1 Supported Tool/OS versions

Tool	Version(s)
RedHat Linux	RHEL4 (WS release 4)
Synopsys VCS	2006.06-SP2-8
Synopsys VERA	2007.12-1
Synopsys CoreTools	B-2008.06-SP2
Synopsys Design Compiler	2007.12-SP3
GNU binutils	2.9
gcc	3.4.6
gmake	3.80
Tcl/Tk	8.3 or 8.4
Perl	5.8.2

2.1.4 Synopsys Verification IP Version (VIP)

The testbench included with the MIPS BusBridge 3 Modules requires the installation of the Synopsys Designware VIP packages listed below. They can be downloaded from the Synopsys website.

- `dw_vip_amba_5.20b.run`
- `dw_vip_ocp_vrt_1.50a.run`

This package should be installed in `$DESIGNWARE_HOME`.

2.2 Installing a Release

Detailed instructions for retrieving, unpacking, and installing the tar file from MIPS, containing deliverables associated with the MIPS BusBridge 3 Modules, were provided when notification about the availability of the release was received.

2.2.1 Unpacking Release Deliverables

The tar file containing deliverables associated with the MIPS BusBridge 3 Modules should be unpacked inside a site-wide installation directory that is accessible to all users. Customers are encouraged to install all MIPS deliverables inside this directory.

In this document, the directory that hosts various installations of MIPS deliverables is referred to as "<mips_home>". Unpacking the tar file in <mips_home> will create a directory tree for this release called <mips_core>. See Section 2.2.2 "Release Naming Convention" for a description of the naming conventions used for MIPS cores .

2.2.2 Release Naming Convention

The MIPS BusBridge 3 Modules release is named in the following format: <core_type>_<release_id>. The <core_type> field is described below:

mbb3 for the MIPS BusBridge 3 Modules release.The <release_id> is in the format x_y_z, with the following meaning:

- x_y refers to the version of the MIPS BusBridge 3 Modules contained in the release. The x is a number indicating the major release version, and y is an alphanumeric value indicating the minor release version; y may be a number, or optionally a number followed by a letter, where the letter indicates a patch update to a minor version number.

- z is a number that refers to the version of all the other supporting deliverables (documentation, BFM model, synthesis scripts, etc.). There is never a patch letter associated with this version. When an RTL version is initially created, the z version number is reset to 0.

Here are an example of legal MIPS BusBridge 3 version name:

- mbb3_2_0_0 # Release with the MIPS BusBridge 3 Modules RTL version 2.0, and version 0 release of other deliverables

2.2.3 Configuring Perl Scripts

Generally, all Perl scripts require perl5. The first line of the Perl scripts included in a release contains the following interpreter line for the location of perl5 used at MIPS:

```
#! /usr/local/bin/perl
```

Since it cannot be assumed that the customer's Perl location is in the same place as MIPS, all of the Perl scripts included in a release must be updated to point to the customer's perl5 location. This is done by using a provided script:

```
% setenv MIPS_HOME <mips_home>
% setenv MIPS_CORE <mips_core>
```

```
% cd $MIPS_HOME/$MIPS_CORE
% perl bin/perl_path.pl <path to perl 5>
```

where the variable <path to perl 5> is the absolute path to the perl5 executable. Note that the `perl_path.pl` script is invoked using the format 'perl <script name>'.

For example, if the perl5 executable is called `perl`, and located in the `/bin` directory, the script would be called with the following parameter:

```
% perl bin/perl_path.pl /bin/perl
```

To find the absolute path to your Perl executable, execute:

```
% which perl
```

Note that the `perl_path.pl` script will verify that the given perl5 location is actually running version 5 of Perl.

2.2.4 Creating a Project Area

While the basic installation of a MIPS BusBridge 3 Modules deliverables is done in the `<mips_home>/<mips_core>` hierarchy, all customizations required in implementing a core should be performed in a separate project directory. The top level directory that contains all deliverables that might require customization is referred to as `<mips_project>`. Before calling the `CreateProject` script to create a new `<mips_project>` directory, *`MIPS_HOME`* and *`MIPS_CORE`* environment variables should be set. If these variables are not defined, the script tries to infer them from the path to the `CreateProject` script. The inferred variables are printed out and should be reviewed by the implementor before proceeding further.

To create a `<mips_project>` directory tree, execute:

```
% setenv MIPS_HOME <mips_home>
% setenv MIPS_CORE <mips_core>
% mkdir <mips_project>
% cd <mips_project>
% <mips_home>/<mips_core>/bin/CreateProject
```

2.2.5 Environment Variable Setup

Once the tar file containing the MIPS BusBridge 3 Modules deliverables has been installed in `<mips_home>/<mips_core>` and the `<mips_project>` directory has been created, there are a number of environment variables that should be set. These are used by the various scripts to configure them for the customer site. A file, `<mips_project>/proc/bin/source.me`, has been included that sets the appropriate variables. Some of these will need to be customized to match the customer site. The environment variables that need to be set are summarized in Table 2.2.

Table 2.2 List of Environment Variables Set in the source.me File

Variable Name	Description	Legal Values
`MIPS_HOME`	Central installation directory for MIPS releases, i.e. `<mips_home>`	Full path of the central installation directory
`MIPS_CORE`	Name of the release, i.e. `<mips_core>`	Name of the release, m*`<core_type>_<release_id>`*

Table 2.2 List of Environment Variables Set in the source.me File (Continued)

Variable Name	Description	Legal Values
MIPS_PROJECT	Top directory containing customized files, i.e., <mips_project>,	Full path to the project directory
MIPS_SITE MIPS_PROFILE	Used to disable internal MIPS scripts	customer
MIPS_PLATFORM	Hosttype	Linux
MIPS_SIM_TYPE	Simulator Type	vcs
MIPS_VCS_VERSION	VCS VERSION	2006_06
The following are not set in the source.me file. These will need to be set by the customer.		
GCC_HOME	Path to GCC install	Path to gcc
VCS_HOME	Path to VCS install directory	Path to VCS install - standard variable for VCS
VERA_HOME	Path to VERA install directory	Path to VERA install - standard variable for VERA
OVL_HOME	Path to Accellera OVL2.3 install directory	Path to Accellera OVL2.3 install directory
DESIGNWARE_HOME	Path to Synopsys DesignWare Install	This is where Synopsys DesignWare IP and VIP packages are installed
DW_VIP_DIR	Path to where DesignWare VIP setup was run.	This directory contains AHB verification IP models.
VMT_VERSION	This indicates the VMT version being used.	This should be set to 3.10a
VIP_AMBA_VERSION	This indicates the AMBA VIP version being used.	This should be set to 5.20b
SYNOPSYS	Path to Synopsys Design Compiler install	Path to Synopsys Design Compiler install - standard variable for Design Compiler

2.2.6 Path Setup

Add the following to your path in order to find scripts and executables used during various steps in implementing a MIPS BusBridge 3.

The following are added to your path in the <mips_project>/proc/bin/source.me script.

- <mips_home>/<mips_core>/bin
- <mips_home>/<mips_core>/flow/bin
- <mips_home>/<mips_core>/flow/verif/bin

The following paths need to be added manually by the customer.

- $SYNOPSYS/bin, $SYNOPSYS/linux/syn/bin
- $DESIGNWARE_HOME/bin
- $VCS_HOME/bin

- $VERA_HOME/bin
- $VERA_HOME/lib
- $GCC_HOME/bin
- Path to make
- Path to perl

2.2.7 Welcome! - The Place to Start

Included in the release is a Welcome file, in HTML format. This file includes information on MIPS support, helpful links, and most importantly, hooks to easily locate documentation for the MIPS BusBridge 3 Modules. It has been designed as the initial reference point for your MIPS BusBridge 3 Module questions. Use your favorite web browser to look at the file $MIPS_HOME/$MIPS_CORE/welcome.html.

2.2.8 Verifying the Release

In order to be sure the release has been delivered and installed properly, users can compile the design and testbench using VCS. Please ensure that the environment variables and path setup described in the previous sections is complete before attempting to proceed to the instructions described below. The following steps must be performed before a VCS compile can be performed.

1. Configure the design. Please edit the configuration file "mbb_config.vh" located in the directory $MIPS_PROJECT/proc/config/customer. The configuration choices are:

    1. For the OCP-SPL module:

    - MBB_SPL: This module facilitates the creation of a dual AXI bridge (OCP-AXI2). The choices are:

      mbb_spl: Use OCP-SPL to create a Dual AXI bridge (OCP-AXI2)

      mbb_spl_stub: Use a stub module for OCP-SPL. This is used when simulating a single AXI bridge (OCP-AXI)

    - MBB_SPL_ADDR_DEC_MODULE: OCP-SPL address decode module. The choices are:

      mbb_spl_addr_dec: The reference address decode module.

      or

      mbb_spl_addr_dec_custom: Customer defined address decode module.

    2. For the OCP-AXI or OCP-AXI2 modules:

    - MBB_SIDEBAND_WIDTH: Width of Sideband signals. See Section 3.1.6.4 "Sideband Signals".
    - MBB_O2A_MCONNID_WIDTH: Width of *MConnID* signal. See Section 3.1.6.4 "Sideband Signals".

    3. For the AXI-OCP module:

- MBB_O2A_MCONNID_WIDTH: Width of *MConnID* if connecting to the IO Coherence Unit (IOCU)
- MBB_O2A_MREQINFO_WIDTH: Width of *MReqInfo* if connecting to the IO Coherence Unit (IOCU)

4. Configuration choices common to all modules:

- MBB_CREGW_MODULE: Fine-grained Clock gating. The choices are:

  mvp_mbb_cregister_gc: Fine grained clock gating enabled (default).

- MBB_ADDR_WIDTH: Width of the MBB3 address bus. Valid values are 32 or 40 only.
- MBB_TAGID_WIDTH: Default width of the *OC_MTagID* bus. Valid values are 4 and 12. This default value can be overwritten by passing a parameter value when instantiating a copy of the MBB3 module. For example:

```
mbb_ocp2axi #(.BUS_ID_WIDTH(4)) my_mbb_ocp2axi (...
```

- MBB_BUS_DEFTYPE: Default bus width or type of the data bus in MBB3. Valid values are 1, 2 or 4. "1" indicates a 64b wide bus, "2" indicates a 128b wide bus, and "4" indicates a 256b wide bus. This default value can be overwritten by passing a parameter value when instantiating a copy of the MBB3 module. For example:

```
mbb_ocp2axi_dual #(.BUS_TYPE(2), ...
```

- mvp_mbb_cregister_ngc: Fine-grained clock gating disabled.
- MBB_NUM_SCAN_CHAIN: Number of Scan chains.

2. Setup Synopsys Verification IP (VIP). Create a directory where you want to setup the VIP. The environment variable DW_VIP_DIR should be set to that directory. Please ensure that you set VMT_VERSION and VIP_AMBA_VERSION as described in Table 2.2.

```
% setenv VMT_VERSION 3.10a

% setenv VIP_AMBA_VERSION 5.20b

% mkdir <vip_install_dir>

% cd <vip_install_dir>

% setenv DW_VIP_DIR $PWD

% dw_vip_setup -path . -vmt $VMT_VERSION -add axi_port_monitor_vmt -v
$VIP_AMBA_VERSION -ntb

% dw_vip_setup -path . -vmt $VMT_VERSION -add axi_interconnect_vmt -v
$VIP_AMBA_VERSION -ntb

% dw_vip_setup -path . -vmt $VMT_VERSION -add axi_slave_vmt -v $VIP_AMBA_VERSION
-ntb

% dw_vip_setup -path . -vmt $VMT_VERSION -add axi_master_vmt -v $VIP_AMBA_VERSION
-ntb

% dw_vip_setup -path . -vrt 1.20a -add ocp_slave_svt -v 1.50a
```

```
% dw_vip_setup -path . -vrt 1.20a -add ocp_monitor_svt -v 1.50a
```

3. Create a configuration file for the Synopsys AMBA VIP models.

   Create a file named “axi_bridge_iconnect.config” under $DW_VIP_DIR. It should contain the following lines:

```
program AxiSlaveProgram.vrp
program AxiPortMonitorProgram.vrp
program AxiInterconnectProgram.vrp
program AxiMasterProgram.vrp
```

4. Build a simulation model (RTL + Testbench) using VCS

```
% cd $MIPS_PROJECT

% mkdir <build_dir>

% cd <build_dir>

For the OCP-AXI bridge or the OCP-AXI2 reference design do

% buildSim -b sa_rtl -local -define OCP2AXI

For the AXI-OCP bridge do

% buildSim -b sa_rtl -local -define AXI2OCP
```

2.3 Directory Structure of the Deliverables

The directory structure of a MIPS BusBridge 3 Modules distribution is shown in Figure 2.1. The contents are divided into two separate directory trees based on whether or not they require modification by the customer. The $MIPS_HOME/$MIPS_CORE hierarchy contains all ‘golden’ components that should only be used as delivered. The $MIPS_PROJECT hierarchy contains components that can be customized if required.

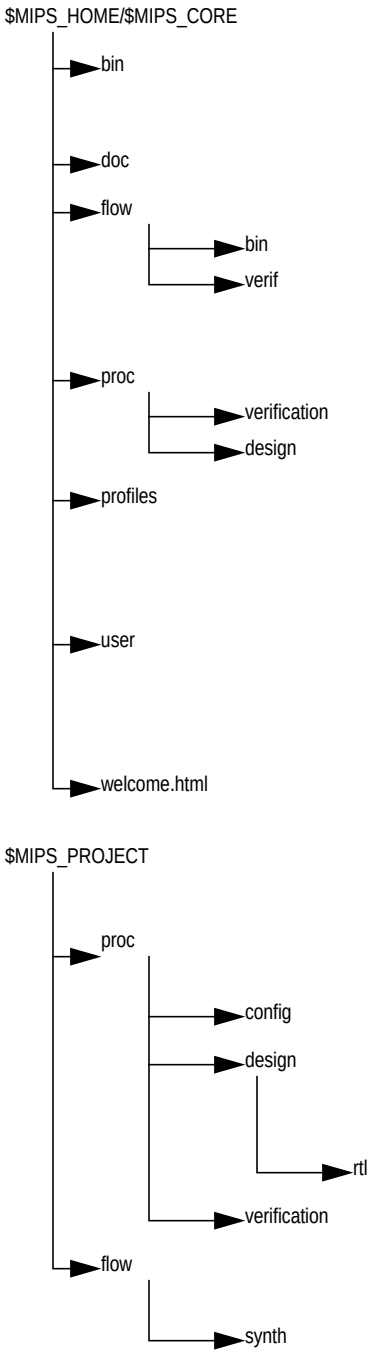


Figure 2.1 Directory Structure of the Bus Bridge 3 Modules

2.3.1 $MIPS_HOME/$MIPS_CORE Directory Tree

The `$MIPS_HOME/$MIPS_CORE` tree contains top-level directories listed in the following subsections. Each subsection briefly describes the contents of each of these directories, and the remainder of this document goes into further detail about using the deliverables contained within them:

- Section 2.3.1.1 "bin Subdirectory"
- Section 2.3.1.2 "doc Subdirectory"
- Section 2.3.1.3 "flow Subdirectory"
- Section 2.3.1.4 "proc Subdirectory"
- Section 2.3.1.5 "profiles Subdirectory"
- Section 2.3.1.6 "user Subdirectory"

2.3.1.1 bin Subdirectory

The `bin` subdirectory contains scripts and/or executables that are needed when creating, simulating or synthesizing the MIPS BusBridge 3 Modules. This directory should become part of the user's path in order to allow access to the scripts as described in Section 2.2.6 "Path Setup".

A MIPS BusBridge 3 Modules release includes the following Perl scripts or executables:

- `buildSim` - Script to create a simulation executable of the core.
- `checksum_deliv` - Script to check that the entire release directory structure was delivered and installed properly.
- `CreateProject` - Creates the `<mips_project>` directory where the core can be configured and customized.
- `perl_path.pl`- Script to update interpreter line of perl scripts to customer's perl version 5 location.

Generally, all Perl scripts require perl5. See Section 2.2.3 "Configuring Perl Scripts" for instructions on how to update all of the perl scripts contained in the release to point to your location of perl5.

2.3.1.2 doc Subdirectory

The `doc` subdirectory contains general documents included in the delivery. The formats may be one of:

- .html - WWW Browser format
- .pdf - Adobe Acrobat format
- .txt - ASCII text format

Documents in this area may be accessed via the file `$MIPS_HOME/$MIPS_CORE/welcome.html`, or directly from the `doc` subdirectory.

A MIPS BusBridge 3 Modules release includes the documents shown in Table 2.3.

Documents that are assigned a MIPS Document number are named with an amalgamation of the document number, security class, short title, document type, and revision; see “MD Document Filename format” on page 27 for more details.

Table 2.3 Description of Documents Found in the doc Subdirectory

MIPS Document Number	Title/Description
MD00660	MIPS® BusBridge™ 3 Modules Users Manual

MD Document Filename format

The filename of all pdf documents with an MD number is in the following format:

```
<MD number>-<security class>-<product>-<type>-<revision>.pdf
```

The fields have the following meaning:

- `<MD number>` consists of the prefix MD and a unique 5 digit number
- `<security class>` defines the confidentiality level of the document. The meaning and treatment of the confidentiality levels is described in a MIPS legal contract. This field is always two characters and usually has one of the following values:
  - 1B: restricted confidential document
  - 1C: internal confidential document
  - 1D: external confidential document
  - 2B: commercial document
- `<product>` is a variable length string holding a descriptive name about the MIPS product to which the document is related.
  - PLATFORMS: for MIPS32 CPU platform specific documents
- `<type>` is a 3 character string that encodes the type of document. Some examples are:
  - USM: users manual
  - AFP: architecture document for programmers
  - APP: application note
  - DTS: data sheet
  - ERS: errata sheet
  - IMG: implementation guide
  - ING: integration guide

- SUM: software user's manual
- SPC: specification

- `<revision>` is a 5 character string in the form *xx.yy*, holding the major (*xx*) and minor (*yy*) revision number of a document.

Since the MD naming scheme is somewhat difficult to decipher, symbolic links have been created with slightly more meaningful names.

2.3.1.3 flow Subdirectory

The `flow` subdirectory contains scripts for synthesis and verification of the MIPS BusBridge 3 Modules and is further subdivided into two subdirectories:

bin Subdirectory

This subdirectory contains the various Perl scripts used while carrying out verification, synthesis, and static timing analysis on the MIPS BusBridge 3 Modules.

`verif` Subdirectory

This subdirectory contains a make subdirectory which contains the common simulator makefile to build the simulation executables.

2.3.1.4 proc Subdirectory

This directory contains the RTL files for the MIPS BusBridge 3 Modules in the following subdirectories:

design/rtl Subdirectory

The `design/rtl` subdirectory contains the majority of Verilog RTL source code of the MIPS BusBridge 3 Modules for synthesis and simulation. Two types of files are present:

- `*.v` files - Verilog RTL source of the MIPS BusBridge 3 Modules for synthesis and simulation.
- `*.vh` files - The include files containing ‘define statements used in RTL files. These files are ‘included into all Verilog files that use the ‘defines.

verification Subdirectory

The `verification` hierarchy contains all files needed to simulate a variety of testbenches provided with the MIPS BusBridge 3 Modules. Notable files or directories in this tree include:

- `testbenches` - Directory holding testbench files
- `build` - Directory holding files related to simulator builds.

2.3.1.5 profiles Subdirectory

The `profiles` subdirectory contains default configurations for `rundiags` and `buildSim`.

2.3.1.6 user Subdirectory

The `user` subdirectory contains files that are used by the `CreateProject` script to construct a fresh `<mips_project>` directory. Files in this directory should not be used for synthesis and simulation and the deliverables provided with the BusBridge 3 Modules do not use these files directly. The script `CreateProject` copies over these files into the `<mips_project>` directory and the BusBridge 3 deliverables pick up these files from there. It is anticipated that users might need to change some of the information in these files (as explained in other portions of this document) and users should be careful to modify these files only in their `<mips_project>` directory.

2.3.2 $MIPS_PROJECT Directory Tree

The `$MIPS_PROJECT` tree contains top-level directories listed in the following subsections. Each subsection briefly describes the contents of each of these directories, and the remainder of this document goes into further detail about using the deliverables contained within them:

- Section 2.3.2.1 “proc Subdirectory”

2.3.2.1 proc Subdirectory

This directory contains files needed for configuring and customizing the MIPS BusBridge 3 Modules. This is also where the simulation executable for the design should be built. The subdirectories are described in more detail below.

bin Subdirectory

The bin subdirectory contains the `source.me` file to set up various environment variables required to simulate or synthesize the design. This file should be customized according to environment of the customer and then sourced before carrying out simulation or synthesis.

config Subdirectory

This directory contains `mbb_config.vh` that is used to configure the MIPS BusBridge 3 RTL.

design *Subdirectory*

This directory contains user modifiable RTL files for the MIPS BusBridge 3.

verification *Subdirectory*

The `verification` subdirectory is where the simulation executable or script is built, as appropriate for the implementor’s chosen simulator, via a standard Makefile.

2.3.2.2 flow Subdirectory

This directory contains the files needed for running synthesis using Synopsys Design Compiler (DC) on the MIPS BusBridge 3 Modules. For a more detailed description on the files, refer to Chapter 7, “Synthesis” on page 85.

`synth/projsyn/lib` Subdirectory

This directory contains the library specific setup file for DC, called `lib_setup.dc`, for running synthesis.

`synth/projsyn/script` Subdirectory

This directory contains all the scripts required to run synthesis on Layer 1 of the MIPS32 CPU platform.

`synth/projsyn/sdc` Subdirectory

This directory contains sdc constraints.

Chapter 3

Functional Descriptions

This chapter describes the basic functionality of the different component blocks of the BusBridge™ 3 Modules. It is organized as follows:

1. Section 3.1 “OCP-AXI Functional Description”
2. Section 3.2 “OCP-SPL Functional Description”
3. Section 3.3 “OCP-AXI2 Functional Description”
4. Section 3.4 “AXI-OCP Functional Description”

3.1 OCP-AXI Functional Description

3.1.1 Functional Block Diagram

The block diagram for the OCP-AXI bridge is shown in Figure 3.1. The bridge consists of the following main components, which are described in more detail in the following sections:

1. Section 3.1.2 “Output Ports”
2. Section 3.1.3 “Pending Read Buffer”
3. Section 3.1.4 “Pending Write Buffer”
4. Section 3.1.5 “OCP 2.1 to AXI 1.0 Map Block”

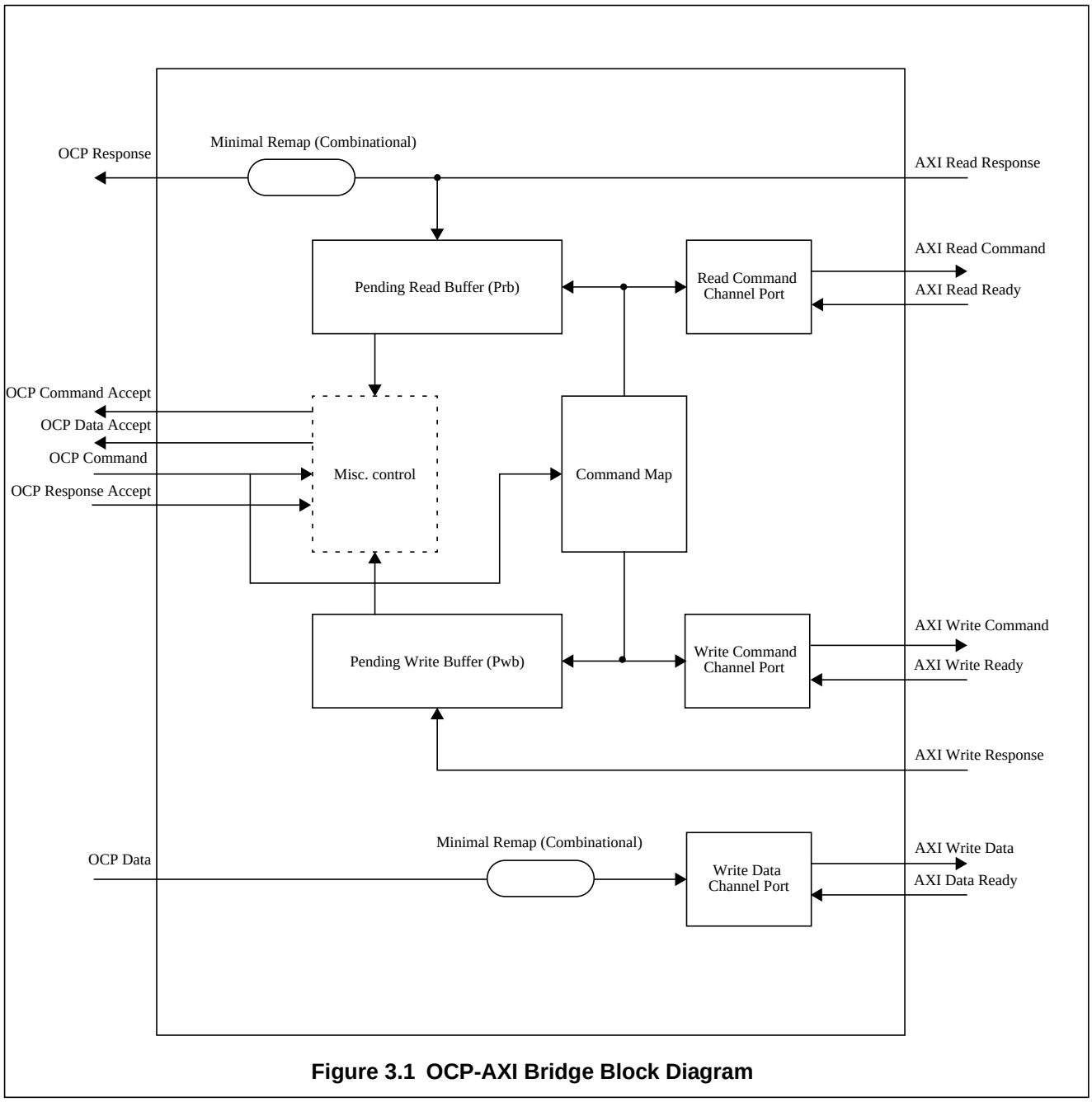


Figure 3.1 OCP-AXI Bridge Block Diagram

3.1.2 Output Ports

The bridge has three output ports to the AXI interface; one each for the read address channel, write address channel, and write data channels. The three output ports are identical from a logic point of view, with the only difference being in the port width. The output ports are used for the following reasons:

1. Eliminate any dependency between the OCP flow control signals and the AXI flow control signals, i.e *OC_SCmdAccept* and *OC_SDataAccept* are not derived from *ARREADY*, *AWREADY*, and *WREADY*.

2. Buffer a stalled OCP transaction to allow other transactions behind it to proceed.

As shown in Figure 3.2, the port consist mainly of a pending register, an output register, a mux to control the data to the output register, and some associated control.

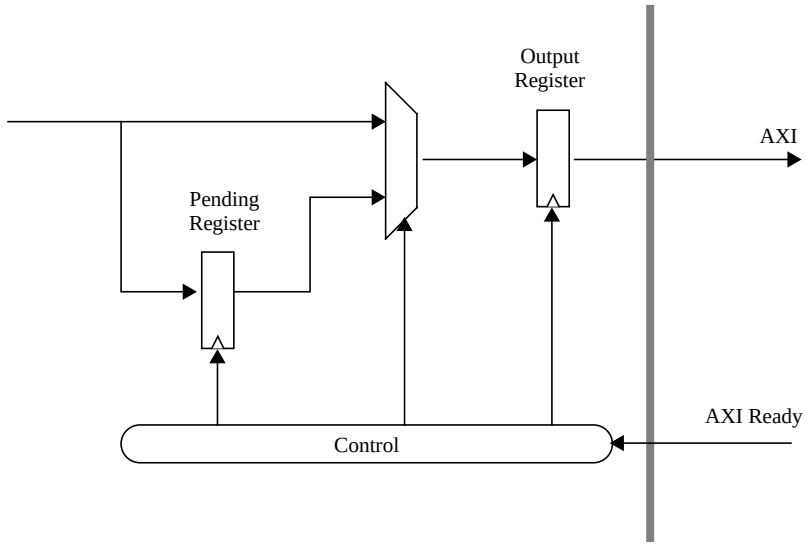


Figure 3.2 Output Port Diagram

3.1.3 Pending Read Buffer

The Pending Read Buffer (Prb) is used to prevent write-after-read hazards (WAR). It consist of 16 entries to keep track of pending read transactions. Each read request that is issued to the read command port is entered into the Prb. The address of the request as well as the burst length are recorded. When the last read data for the request is received by the Prb, the corresponding entry is deleted. Note that the Prb is tag compared and selected by the ID of the read request.

When a write request is getting issued to the bridge, its address is checked against all pending reads. If there is an overlap, then a Prb hit is signal and the matching entries are recorded in a match vector. The write request is then prevented from updating the output register. Instead, it is staged in the pending register in the write command port and waits until all matching read entries completed. During that time, no new write requests are accepted by the OCP-AXI bridge. Note that write data and read requests can still be accepted when the write request is stalled due to WAR hazard.

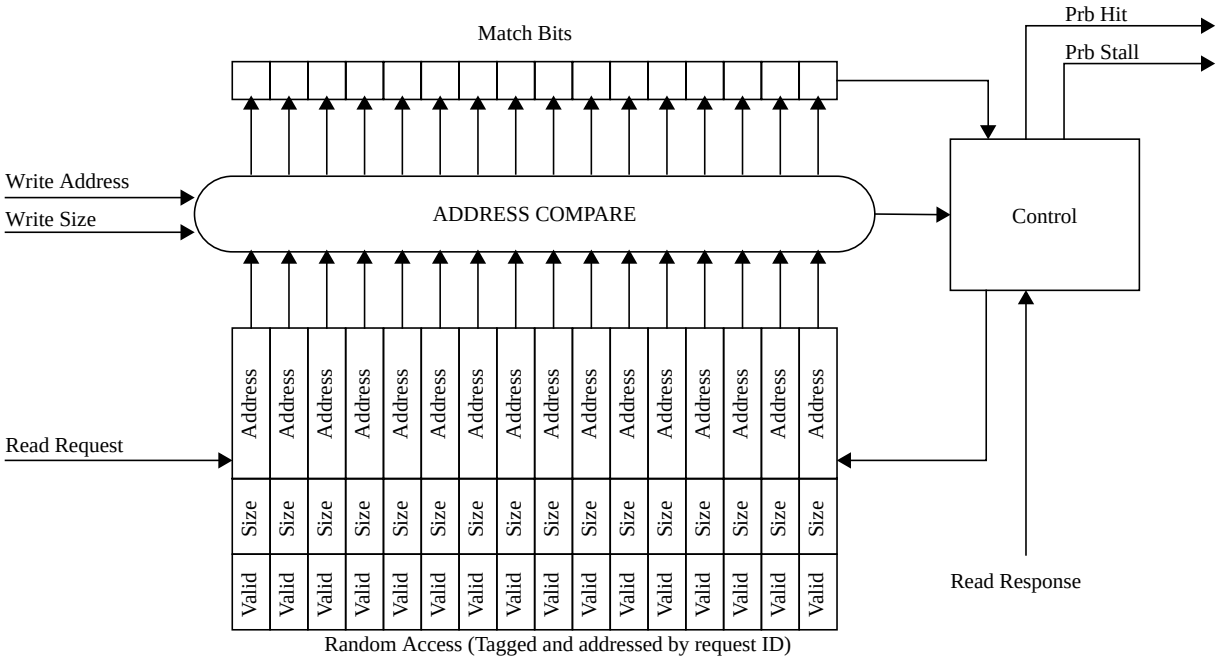


Figure 3.3 Pending Read Buffer

3.1.4 Pending Write Buffer

The Pending Write Buffer (Pwb) is used to prevent read-after-write hazards (RAW). It consist of 8 entries to keep track of pending write transactions. Each write request that is issued to the write command port is entered into the Pwb. The address of the request as well as the burst length are recorded. When the completion response for the write is received by the Pwb, the corresponding entry is deleted. The Pwb is analogous in functionality to the Prb, but is structured as a FIFO, because all writes complete in order.

When a read request is issued to the bridge, its address is checked against all pending writes. If there is an overlap, then a Pwb hit is signaled, and the matching entries are recorded in a match vector. The read request is then prevented from updating the output register. Instead, it is staged in the pending register in the read command port, and waits until all matching write entries completed. During that time, no new read requests are accepted by the OCP-AXI bridge. Note that write data and write requests can still be accepted when the read request is stalled due to WAR hazard. When there is no available Pwb entry, the bridge does not accept any more write commands and backpressures the OCP master.

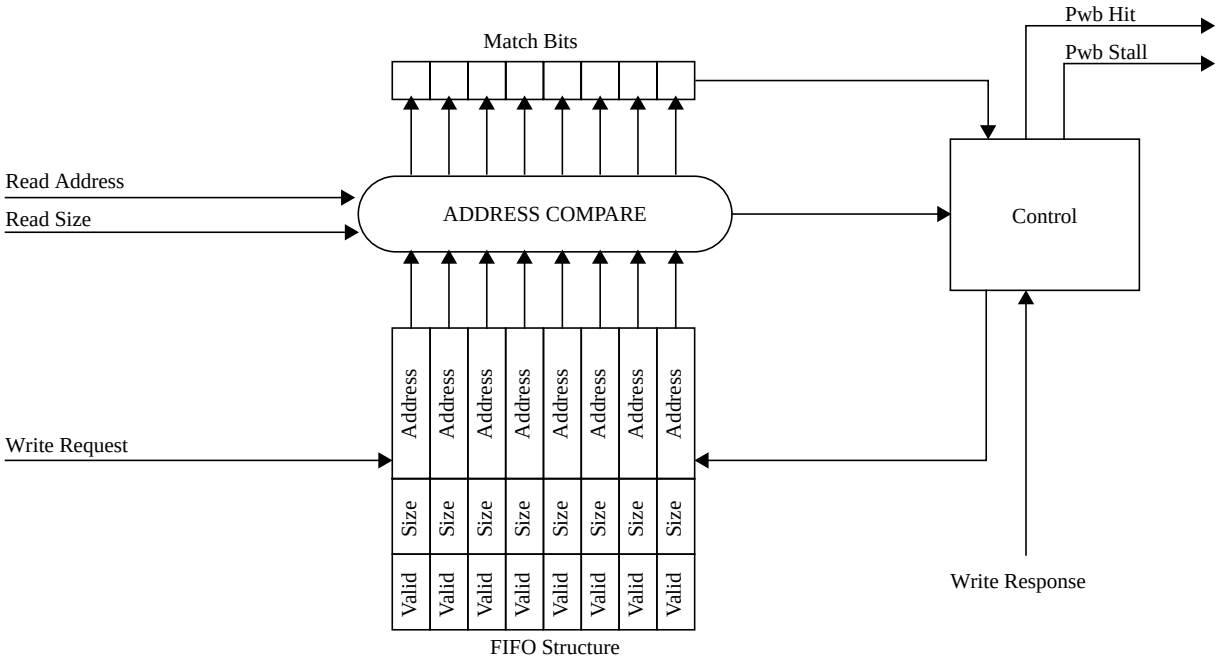


Figure 3.4 Pending Write Buffer

3.1.5 OCP 2.1 to AXI 1.0 Map Block

The command mapping block is a combinational block that maps OCP encodings to AXI encodings. The detailed mapping are described in the following sections.

3.1.5.1 OCP 2.1 Request Signals

OC_MCmd[3:0]

OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on *OC_MReqInfo* and *OC_MAddrSpace*. The encodings used are shown in the following table:

Table 3.1 OC_MCmd Mapping

Encoding	Command	Mnemonic	AXI Mapping	Notes
0	Idle	IDLE	None	No transaction
1	Write	WR	AWVALID	L2 cacheop writes are not mapped

Table 3.1 OC_MCmd Mapping (Continued)

Encoding	Command	Mnemonic	AXI Mapping	Notes
2	Read	RD	ARVALID	L2 cacheop reads are not mapped
3-7	Unused	-	-	Not used

OC_MReqInfo[6:0]

For transactions other than SYNC and CACHE, the *OC_MReqInfo[2:0]* field encodes the cacheability attributes for a transaction; it uses the same encoding as the CCA field described in a *MIPS32® Processor Core Family Software Users Manual*. *OC_MReqInfo[3]* indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.

The encoding of the *OC_MReqInfo* field for all transactions other than CACHE is summarized in the following table:

Table 3.2 OC_MReqInfo Mapping

Encoding	Command Information	AXI Mapping (ARCACHE[3:0]) (WA = write allocate, RA= read allocate, C= cacheable, B = bufferable)			
		WA	RA	C	B
0	Cacheable, noncoherent, WT, NWA	0	1	1	0
1	Cacheable, noncoherent, WT, WA	1	1	1	0
2	Uncached, noncoherent	0	0	0	0
3	Cacheable, noncoherent, WB, WA	1	1	1	0
4	Cacheable, coherent, WB, WA, exclusive	1	1	1	0
5	Cacheable, coherent, WB, WA, exclusive on write	1	1	1	0
6	Reserved	0	0	0	0
7	Uncached accelerated	0	0	0	0
8-9	Reserved	0	0	0	0
10	SYNC with uncached CCA	0	0	0	0
11-15	Reserved	0	0	0	0

Note: The AXI bridge will **not** map *OC_MReqInfo[6:4]*, since these deal with L2 cache exclusivity control.

OC_MAddrSpace[1:0]

The *OC_MAddrSpace* signal is used as L2/L3 Address Space indicator. When the core is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.

The AXI bridge does **not** map this field since an L2 or an L3 cache should be on the OCP side.

OC_MAddr['MBB_ADDR_WIDTH-1:0]

The *OC_MAddr* is the physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (*OC_MByteEn*) or write (*OC_MDataByteEn*) byte enable fields. This field maps to:

1. *ARADDR[‘MBB_ADDR_WIDTH-1:0]* (Read requests)

2. *AWADDR[‘MBB_ADDR_WIDTH-1:0]* (Write requests)

Note that on AXI, the address bus refers to the starting byte address, so the *OC_MByteEn* are used to generate the lower address *ARADDR*.

OC_MBurstSeq[2:0]

This field indicates the type of burst sequence. The core can only generate two possible values, determined by the *SI_SBlock* static input, as shown in the following table:

Table 3.3 OC_MBurstSeq Mapping

Encoding	Burst Sequence	AXI Mapping
2	Sequential: Critical dword first, with linear wrapping for subsequent beats	ARBURST[1:0] = WRAP (Read request) AWBURST[1:0] = WRAP (Write request)
4	Sub-block Critical dword first, with increment/decrement for subsequent beats	AXI does not support sub-block ordering. Therefore SI_SBlock **should be tied to 0**
0-1,3,5-7	Unused by 24Kc core	-

OC_MTagID[‘MBB_TAGID_WIDTH-1:0]

The transaction tag identifier maps directly to the read transaction ID *ARID* for read requests, and to the write transaction ID *AWID* for write requests. Note that the field is also used to generate the instruction/data bit indicator *ARPROT[2]* on the AXI interface.

Table 3.4 OC_MTagID[`‘MBB_TAGID_WIDTH-1:0] Mapping

Encoding	Tag Allocation	AXI Mapping
0	From Read buffer 0	ARPROT[2] = 0
1	From Read buffer 1	
2	From Read buffer 2	
3	From Read buffer 3	
4	From Fetch buffer 0	ARPROT[2] = 1
5	From Fetch buffer 1	
6	SYNC	Does not get passed to AXI
7	WR, CACHE-RD, CACHE-WR	-
8-11	Reserved	
12	From Fetch buffer 2	ARPROT[2] = 1
13	From Fetch buffer 3	
14-15	Reserved	-

Table 3.4 OC_MTagID[`‘MBB_TAGID_WIDTH-1:0] Mapping (Continued)

Encoding	Tag Allocation	AXI Mapping
8	From Read buffer 4	ARPROT[2] = 0
9	From Read buffer 5	
10	From Read buffer 6	
11	From Read buffer 7	
12	From Fetch buffer 2	ARPROT[2] = 1
13	From Fetch buffer 3	
14-15	Reserved	-

Note that the correlation of IDs to processor buffers is only valid when the bridge is directly connected to the processor core.

OC_MBurstPrecise

Indicates whether the burst length is precise. Burst lengths are always fixed to either 4 or 8 beats, so this pin is static set to 0x1.

OC_MBurstSingleReq

Indicates whether there is a single request for all data transfers in a burst. In the core, there is always a single command request so this pin is statically set to 0x1. This signal is unused by the AXI bridge.

OC_MBurstLength[2:0]

Number of 64b data transfers, only three values are possible

Table 3.5 OC_MBurstLength Mapping

Encoding	Number of Transfers	AXI Mapping
1	1, single transfer	ARLEN[3:0] = 4‘b0000 (Read, 1 transfer, 64-bit AXI data paths) AWLEN[3:0] = 4‘b0000 (Write, 1 transfer, 64-bit AXI data paths)
4	4-beat burst	ARLEN[3:0] = 4‘b0011 (Read, 4 transfers, 64-bit AXI data paths) AWLEN[3:0] = 4‘b0011 (Write, 4 transfers, 64-bit AXI data paths)
8	8-beat burst	ARLEN[3:0] = 4‘b0111 (Read, 8transfers, 64-bit AXI data paths) AWLEN[3:0] = 4‘b0111(Write, 8 transfers, 64-bit AXI data paths)
[k] (xth bit in OC_MByteEn)	[9*k-1:8*k]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = 'x || (1 << (k + 1)) --> lower-bits of ARADDR = 'd(k)
others	Unused by core	-

OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0]

Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the *OC_MByteEn* field to the returned read data bytes is shown in the following table. Note that the byte enables are used to determine the starting byte address on the AXI side.

Table 3.6 OC_MByteEn Mapping

OC_MByteEn	Requested byte to be returned on *OC_SData* bus	AXI Mapping
[0]	[7:0]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x1 -> ARADDR[2:0] = 3‘b000
[1]	[15:8]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x10 -> ARADDR[2:0] = 3‘b001
[2]	[23:16]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x100 -> ARADDR[2:0] = 3‘b010
[3]	[31:24]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x1000 -> ARADDR[2:0] = 3‘b011
[4]	[39:32]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x10000 -> ARADDR[2:0] = 3‘b100
[5]	[47:40]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x100000 -> ARADDR[2:0] = 3‘b101
[6]	[55:48]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x1000000 -> ARADDR[2:0] = 3‘b110
[7]	[63:56]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = ...x10000000 -> ARADDR[2:0] = 3‘b111
[k]	[9*k-1:8*k]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0] = 'x || (1 << (k + 1)) --> lower-bits of ARADDR = 'd(k) where k is the xth bit in OC_MByteEn

3.1.5.2 Write Data OCP Signals

The write data OCP signals map directly to their AXI counterparts.

Table 3.7 OCP Write Data Bus Mappings

OCP Signal	Description	AXI Mapping
OC_MData	Write data	*WDATA*
OC_MDataByteEn	Byte enables for writes.	*WSTRB*
OC_MDataValid	Valid write data	*WVALID*
OC_MDataTagID	Write data tag identifier	*WID*
OC_MDataLast	Last data in a write burst	*WLAST*

3.1.5.3 OCP 2.1 Response Signals

In this section, we discuss how the AXI response signals are mapped back into OCP. To start off, the read data and tag identifier, and last data indicators map back directly to their OCP equivalents:

Table 3.8 Response Data Mapping

AXI Signal	Description	OCP Mapping
RID	Read ID Tag	*OC_STagID*
RDATA	Read Data	*OC_SData*
RLAST	Read Last	*OC_SRespLast*

The AXI response signal *RRESP[1:0]* maps back to the *OC_SResp[1:0]* and *OC_SRespInfo[1:0]* as follows:

Table 3.9 RRESP Mapping

AXI Signal	Description	OCP Mapping
RRESP[1:0] == 2‘b00	OKAY	OC_SRESP[1:0] = 2‘b01 (Data valid)
RRESP[1:0] == 2‘b01	EXOKAY	N/A (AXI bridge does not make exclusive accesses)
RRESP[1:0] == 2‘b10	SLVERR	OC_SResp[1:0] = 2‘b11 (Error), OC_SRespInfo = 2‘b00 (Bus Error)
RRESP[1:0] == 2‘b11	DECERR	OC_SResp[1:0] = 2‘b11 (Error), OC_SRespInfo = 2‘b00 (Bus Error)

3.1.5.4 Unmapped Signals

There remains some signals on AXI side that are not mapped from OCP. Those assume some default values and are listed in this section.

AXI Write Response Signals

The OCP interface on MIPS cores does not use write responses. So all the write response signals from AXI are not passed through. Those include the following: *BID*, *BRESP*, *BVALID*, and *BREADY*. The AXI write responses are used internally in the bridge for SYNC handling as well as RAW (read-after-write) hazard handling.

AWLOCK and ARLOCK Handling

The AXI bridge will not make any locked or exclusive accesses, so these signals are hardwired to normal access value, i.e.: *AWLOCK[1:0]* = 2‘b00, *ARLOCK[1:0]* = 2‘b00.

ARPROT and AWPROT Handling

These signals are used to communicate some additional information about the request:

- *ARPROT[0]* and *AWPROT[0]*: Normal or privileged access (0 is normal access)
- *ARPROT[1]* and *AWPROT[1]*: Secure or non-secure (1 is non-secure)
- *ARPROT[2]* and *AWPROT[2]*: Instruction or data (0 is data access)_

The only bit that is mapped by the bridge it bit [2] since we can deduce the instruction/data type of the request from the tag identified on the OCP side. Note however that the information is valid only when the bridge is connected directly to the processor core.

The lower two bits are user selectable using the port AXI_CMD_PROT[1:0]. The user can set the value for these statically or through some external logic to the bridge. The bridge associates the AXI_CMD_PROT[1:0] with the current OCP request and passes that information over when it remaps the request to the AXI side.

ARBURST and AWBURST

The burst type on the AXI will be set to one of the following values based on the burst length:

- Burst length of 1: *ARBURST[1:0]* = *AWBURST[1:0]* = 2‘b01 (INCR)
- Burst length of 4 or 8: *ARBURST[1:0]* = *AWBURST[1:0]* = 2‘b10 (WRAP)

ARSIZE and AWSIZE

The burst size will be hardwired to use the full AXI data bus width for burst transfers, i.e:

- *ARSIZE[2:0]* = *ARSIZE[2:0]* = 3‘b011 (8 bytes, for AXI with 64-bit data path)

3.1.6 Special Topics

3.1.6.1 SYNC Handling

The AXI bridge supports externalized SYNC requests. An external SYNC request is actually an OCP read command with the following characteristics:

Table 3.10 SYNC Transaction Signals

Signal	Value for Sync Transaction
OC_MAddr	*OC_MAddr*[*‘MBB_ADDR_WIDTH*-1:8] is always 0x1fc000. *OC_MAddr[7:3]* holds the stype bits [10:6] from the SYNC instruction.
OC_MByteEn	Always 0
OC_MBurstLength	Always 1
OC_MReqInfo[3:0]	Always 0xA *OC_MReqInfo[3]* identifies the SYNC, while *OC_MReqInfo[2:0]* specify an uncached CCA.
OC_MAddrSpace[1:0]	Always 2‘b00 (Normal address space)

When the AXI bridge decodes a SYNC request on the OCP interface, it will perform the most complete set of synchronization operations that are defined. This means the bridge does a completion barrier that affects both load and stores preceding the SYNC instruction and both loads and stores that are subsequent to the SYNC instruction. Specifically, when the bridge detects a SYNC transaction it will:

1. Decline any additional requests from the processor core.
2. Wait until all pending read/write transactions are completed.

3. Issue the read command (SYNC) to the AXI bus. When the response returns, the bridge resumes acceptance of requests from the OCP master.

3.1.6.2 L2/L3 cacheops

The execution of the CACHE instruction, allowing privileged software to manage an L2 or L3 external cache, also results in transactions on the OCP interface. Again, these L2/L3 cache operations are not defined by the OCP interface, and are basically proprietary. However, these operations can only occur when an L2/L3 cache has been enabled, and a CACHE instruction intended for an L2/L3 cache is executed.

An L2/L3 cacheop transaction on the OCP interface is distinguished by a non-zero value of the *OC_MAddrSpace* signal in the following way:

Table 3.11 L2/L3 Cacheop Handling

OC_MAddrSpace[1:0]	Description
0x0	Normal Read/Write or SYNC
0x1	L2 cacheop
0x2	L3 cacheop
0x3	Reserved

The AXI bridge does **not** support L2/L3 cacheops, and the *OC_MAddrSpace* value is effectively ignored. However, the AXI bridge should **never** be positioned between the processor core and the L2/L3 cache, so there should not be a case where the AXI bridge receives an L2/L3 cacheop operation.

If the AXI bridge receives an L2/L3 cacheop transaction, it will be treated as a regular read/write operation since no special decoding of *OC_MAddrSpace* is implemented.

3.1.6.3 Write Errors

Because the cores use posted writes, an error write response cannot be communicated directly back. Instead, when the AXI bridge detects an error write response, it asserts the signal *AXI_WERR_INT*. This signal can be used to generate an interrupt to the processor core. The signal remains asserted until it is cleared by the interrupt acknowledge signal *AXI_WERR_ACK*. There are also two additional signals set by the bridge when *AXI_WERR_INT* is asserted:

1. *AXI_WERR_ADDR['MBB_ADDR_WIDTH-1:0]*: This is the address of the write that completed with error.

2. *AXI_WERR_TYPE*: This signal indicates if the write error is due to a decode or slave error (0 = `SLVERR`, 1 = `DECERR`)

3.1.6.4 Sideband Signals

The bridge includes sideband signals that can be used by system designer to pass additional information from the OCP to AXI domains. Note that these signals are not part of either protocol, but are included for maximum flexibility. The width of the sideband signals are set by the value of '`MBB_SIDEBAND_WIDTH` and '`MBB_O2A_MCONNID_WIDTH` in the configuration file, whereas, the data sideband signals are set by the value of '`MBB_SIDEBAND_WIDTH`. If these signals are not going to be used, it is recommended to connect the input sideband signals to 0.

There are three sideband input buses from the OCP side:

1. *OC_MCmdSideBand*: This sideband bus associates with the current OCP command on the bus (*OC_MCmd*). The bridge records the value on that bus along with the request. The sideband value is then passed along with its corresponding command on the AXI bus on the *AWSIDEBAND* signal if the request is a write, or on the *AERSIDEBAND* signal if the request is a read.

2. *OC_MConnID*: Sideband signal associated with *OC_MCmd*. The bridge records the value on that bus along with the request. The sideband value is then concatenated with the *OC_MCmdSideBand* and passed along with its corresponding command on the AXI bus on the *AWSIDEBAND* signal if the request is a write, or on the *ARSIDEBAND* signal if the request is a read.

3. *OC_MDataSideBand*: This sideband bus associates with the current OCP data on the bus (*OC_MData*). The bridge records the value on that bus along with the data. The sideband value is then passed along with its corresponding data on the AXI bus on the *WSIDEBAND* signal.

The concatenation places *OC_MConnID* in the MSB of *AWSIDEBAND*/*ARSIDEBAND* and places *OC_MCmdSideBand* in the LSB of *AWSIDEBAND*/*ARSIDEBAND*.

3.2 OCP-SPL Functional Description

3.2.1 OCP-SPL Block Diagram

The OCP Splitter is not a standard design. Consult the MIPS softcore product datasheet to determine whether the OCP Splitter is included in the release. If applicable, this section describes the signals used in OCP Splitter.

The block diagram for the OCP-SPL bridge is shown in Figure 3.5. The OCP-SPL consists of the following main components which are described in more detail in the following sections:

1. Section 3.2.1.1 “OCP-SPL Address Decode”

2. Section 3.2.1.2 “OCP-SPL Response FIFOs”

3. Section 3.2.1.3 “OCP-SPL Configuration”

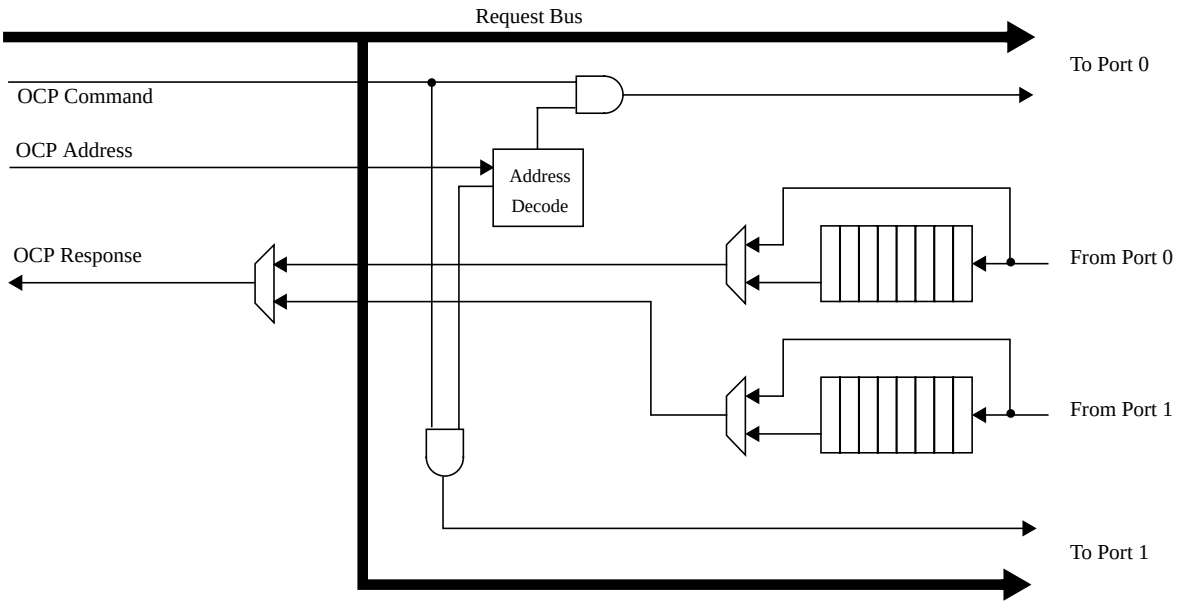


Figure 3.5 OCP-SPL Block Diagram

3.2.1.1 OCP-SPL Address Decode

The OCP-SPL block performs a simple function. It connects an OCP master to two OCP slave ports. OCP requests are directed to either port 0 or port 1 based solely on address decode function. Responses from the two ports are merged back into a single response bus to the OCP master using a priority scheme.

The request path from the OCP master to either port is purely combinational, there is no additional latency incurred. As a result, the address decode function is preferably simple to avoid any timing paths. The OCP-SPL supports two address decode implementations: a reference implementation, and a customer implementation.

The reference address decode allocates the lower 256 MByte address region to port 0, with the remaining address space is allocated to port 1. As a result, the address ranges are as follows:

- Port 0: 0x00000000-0x0FFFFFFF
- Port 1: 0x10000000-0xFFFFFFFF

The reference address decode is used as default, and the customer most likely would chose to implement a different address decode scheme. In this case, there is a file that should be edited to install a new address decode scheme:

`$MIPS_PROJECT/proc/design/rtl/mbb_spl_addr_dec_custom/mbb_spl_addr_dec_custom.v`

The module has one input port, *OC_MAddr['MBB_ADDR_WIDTH-1:0]*, which is the full OCP address, and 1 output port "*target_port*" which indicates the target of the OCP request:

- *target_port* == 1‘b0: Request is for port 0
- *target_port* == 1‘b1: Request is for port 1

The *target_port* is a purely combinational function of *OC_MAddr[‘MBB_ADDR_WIDTH-1:0]*. The customer can chose any arbitrary function provided the implementation meets the customer timing requirements. After editing the file, then the configuration file needs to be edited to specify the custom address decode file:

```
$MIPS_PROJECT/config/customer/mbb_config.vh
```

The following line should be changed from:

```
‘define MBB_SPL_ADDR_DEC_MODULE mbb_spl_addr_dec
```

to

```
‘define MBB_SPL_ADDR_DEC_MODULE mbb_spl_addr_dec_custom
```

3.2.1.2 OCP-SPL Response FIFOs

The OCP-SPL contains two response FIFOs, one for port 0 and one for port 1. Each FIFO has eight 64-bit entries and as a result can store a full 8-beat response burst. The FIFOs are needed in case response data is received simultaneously from both ports. In that case, the response from one port is allowed to proceed to the OCP master and the response from the other port is stored in its corresponding FIFO.

When response data is buffered into the FIFO, it is sent to the OCP master as soon as there is no contention on the response bus from the other port. As a result, the FIFO does not introduce any inefficiency in the response data path. If there is no contention on the OCP response bus, then the response data from either port can bypass the FIFO entirely and can reach the OCP master with no additional latency as shown in Figure 3.5.

When there is contention for the response bus, the selected port response is selected using a fixed priority scheme. The priority function is based on OCP-SPL configuration and is explained in more detail in Section 3.2.1.3 “OCP-SPL Configuration”.

3.2.1.3 OCP-SPL Configuration

The OCP-SPL operation/personality is affected by a number of factors:

1. Address decode: This was covered in Section 3.2.1.1 “OCP-SPL Address Decode”.
2. Response flow control: This indicates to the OCP-SPL if any of the OCP interfaces support response flow control.
3. Default priority: This, along with response flow control configuration, affects how responses are prioritized.

Response Flow Control Configuration

The OCP-SPL block has three response flow control configuration inputs:

1. *OC_MRespAccept_En*: This is a static signal. When it is connected to 1‘b1, it indicates that the OCP master supports response flow control, and that the *OC_MRespAccept* signal is active. If it is connected to 1‘b0, it indicates that the OCP master does not support response flow control. In this case, the *OC_MRespAccept* input **must** be connected to 1‘b1.

2. *P0_MRespAccept_En;* This is a static signal. When it is connected to 1‘b1, it indicates that the OCP slave connected to port 0 support response flow control and uses the *P0_MRespAccept* output signal from the OCP-SPL block. In this case, the OCP-SPL can back-pressure responses from the OCP slave on port 0. When the signal is connected to 1‘b0, it indicates that the OCP slave connected to port 0 does not support response flow control, and as a result the OCP-SPL must be ready to accept any response from port 0.

3. *P1_MRespAccept_En;* This is a static signal. When it is connected to 1‘b1, it indicates that the OCP slave connected to port 1support response flow control and uses the *P1_MRespAccept* output signal from the OCP-SPL block. In this case, the OCP-SPL can back-pressure responses from the OCP slave on port 1. When the signal is connected to 1‘b0, it indicates that the OCP slave connected to port 1does not support response flow control, and as a result the OCP-SPL must be ready to accept any response from port 1.

Default Priority

The OCP-SPL block has a static input that specifies the default priority for response data:

- *dflt_port_priority*: When this signal is set to 1‘b0, it indicates that the default priority is given to port 0. When this signal is set to 1‘b1, it indicates that port 1 has default priority. Note that the effective priority may be different from the default priority, and it is derived from this signal as well as the response flow control configuration signals described in Section “Response Flow Control Configuration”.

Effective Priority And Read Request Restrictions

Since the OCP-SPL has limited buffering capability for response data, read requests may be restricted to one of both ports. This is dependent on the response flow control configuration as described in Section “Response Flow Control Configuration”. In addition, the effective priority may be changed from the default priority based on the flow control signals. The effective priority and the port read request restrictions are determined according to the following table:

Table 3.12 Effective Priority And Read Request Restrictions

OC_AccEn	P0_AccEn	P1_AccEn	Restriction	Priority
0	0	0	1 read to low priority port	default
0	0	1	None	port 0
0	1	0	None	port 1
0	1	1	None	default
1	0	0	1 read for each port	default
1	0	1	1 read to port 0	default
1	1	0	1 read to port 1	default
1	1	1	None	default

Legend

```
OC_AccEn = OC_MRespAccept_En
P0_AccEn = P0_MRespAccept_En
P1_AccEn = P1_MRespAccept_En
```

3.3 OCP-AXI2 Functional Description

3.3.1 OCP-AXI2 Block Diagram

The OCP-AXI2 connects an OCP master to two AXI slave interfaces. As shown in Figure 3.6, it is composed of one OCP-SPL block and 2 OCP-AXI2 blocks. Note that arrow directions in the diagram indicate the request direction. The response arrows flow in the opposite direction and are not shown for simplicity.

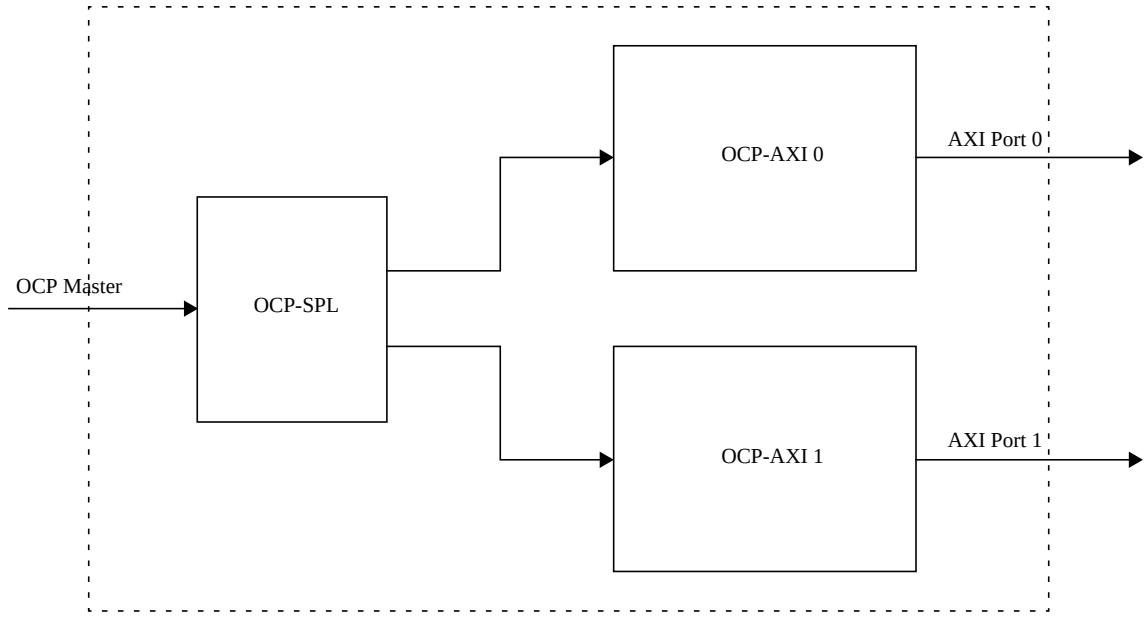


Figure 3.6 OCP-AXI2 Block Diagram

3.3.2 OCP-AXI2 Configuration

The OCP-AXI2 block is provided as a convenient pre-packaged and tested solution. Its functional description is completely described by the functional description of its component blocks. (See Section 3.1 “OCP-AXI Functional Description” and Section 3.2 “OCP-SPL Functional Description”). There are a limited number of parameters you can configure for the OCP-AXI2:

1. Address decode: To determine how requests are routed to port 0 and port 1, and how to customize it, you can refer to Section 3.2.1.1 “OCP-SPL Address Decode” for more detail.

2. Port priority: When responses from both port 0 and port 1 are returning through the splitter, a priority scheme is chosen to determine who gets access to the OCP master response bus first. This is partially controlled with a static input port called `dflt_port_priority`. This is described in detail in Section “Default Priority” and Section “Effective Priority And Read Request Restrictions” in the OCP-SPL functional description.

3. Response Flow Control: The OCP-AXI2 has three response flow control configuration ports. Note that these ports are used only by the OCP-SPL block:

    - *OC_MRespAccept_En*: This is a static signal. When it is connected to 1‘b1, it indicates that the OCP master supports response flow control, and that the *OC_MRespAccept* signal is active. If it is connected to 1‘b0, it indicates that the OCP master does not support response flow control. In this case, the *OC_MRespAccept* input **must** be connected to 1‘b1.

    - *P0_MRespAccept_En:* This is a static signal and **must** be connected to 1‘b1. It is provided as a port only for internal testing purposes of OCP-SPL block. Connecting this port to 1‘b0 may result in performance loss.

    - *P1_MRespAccept_En:* This is a static signal and **must** be connected to 1‘b1. It is provided as a port only for internal testing purposes of OCP-SPL block. Connecting this port to 1‘b0 may result in performance loss.

3.4 AXI-OCP Functional Description

The AXI-OCP bridge is intended to allow an AXI subsystem in an SOC to connect to the following OCP slave interfaces in MIPS32 cores:

- ScratchPad RAM (ISPRAM/DSPRAM) DMA interfaces on the 24K, 34K, 74K and 1004K cores
- I/O Coherence Unit (IOCU) in the 1004K Coherent Processing System

The block diagram for the AXI-OCP bridge is shown in the Figure 3.7. The bridge is composed of the following sub-blocks, which are described in the following sections:

1. Section 3.4.1 “AXI Command and Data Ports”
2. Section 3.4.2 “AXI Command Map”
3. Section 3.4.3 “OCP Port”
4. Section 3.4.4 “Pending Transaction FIFO”

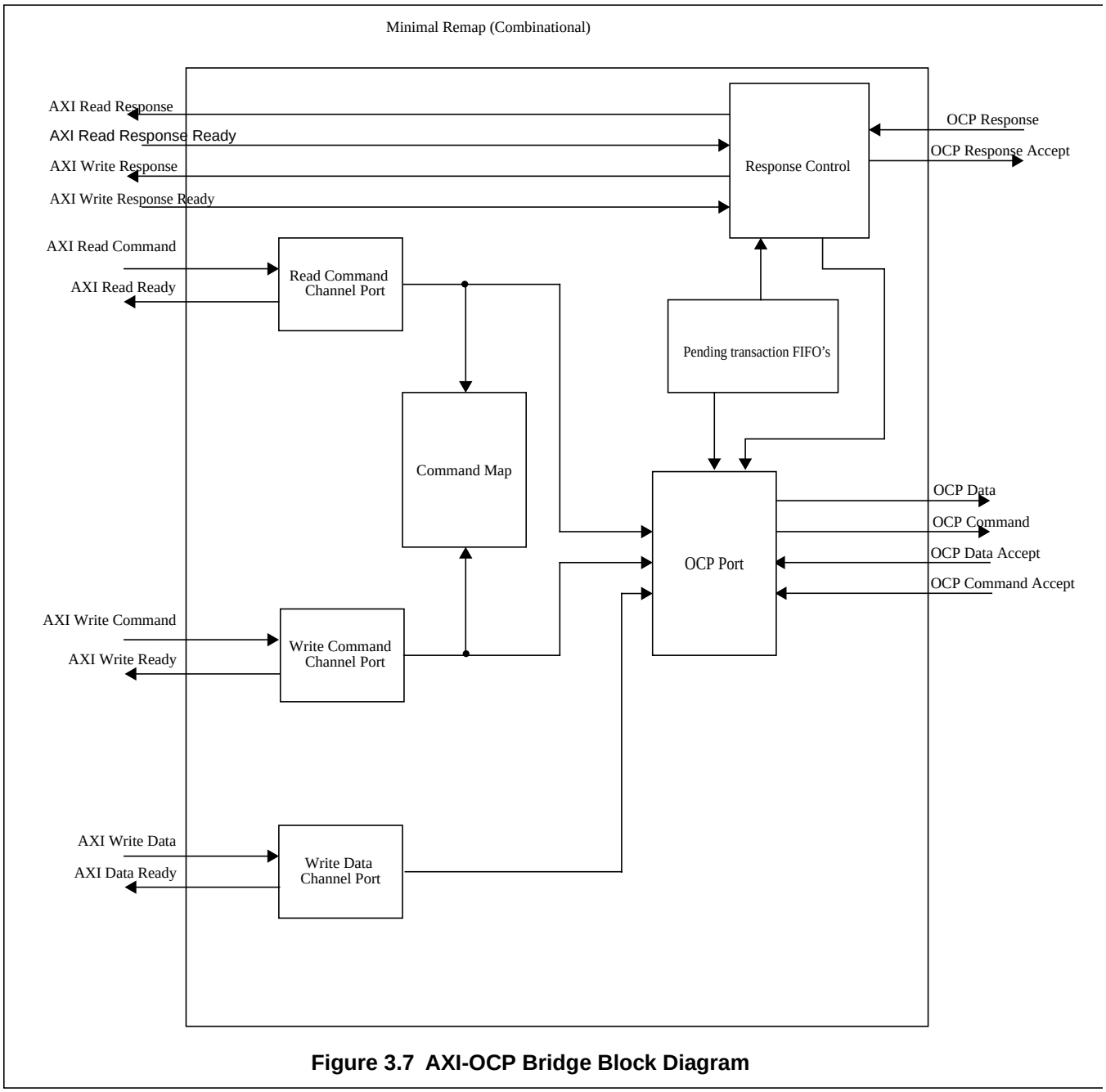


Figure 3.7 AXI-OCP Bridge Block Diagram

3.4.1 AXI Command and Data Ports

The bridge has 3 input ports on the AXI interface - Read address channel, Write address channel and the Write data channel. The Read and Write address channels are identical, while the Write data channel is slightly different since it has to make sure that the Write data is sent on the OCP port only after the Write command phase on the OCP port. This is due to that fact that the AXI Write address and data channels are independent and have no timing relationship with each other, but the OCP protocol does not allow for the Write data phase to precede the Write Command phase.

The input ports are used for the following reasons:

1. Eliminate any dependency between the OCP flow control signals and the AXI flow control signals, i.e., *ARREADY*, *AWREADY*, and *WREADY* are not combinationally derived from *OC_SCmdAccept* and *OC_SDataAccept*. The input ports also allow us to decouple the relationship between the VALID and READY signals on the AXI channels.

2. Provide the ability to buffer the write data when it arrives before the write command.

As shown in Figure 3.8, the port consists mainly of a pending register, a mux to control the data to the OCP port sub-block, and some associated control.

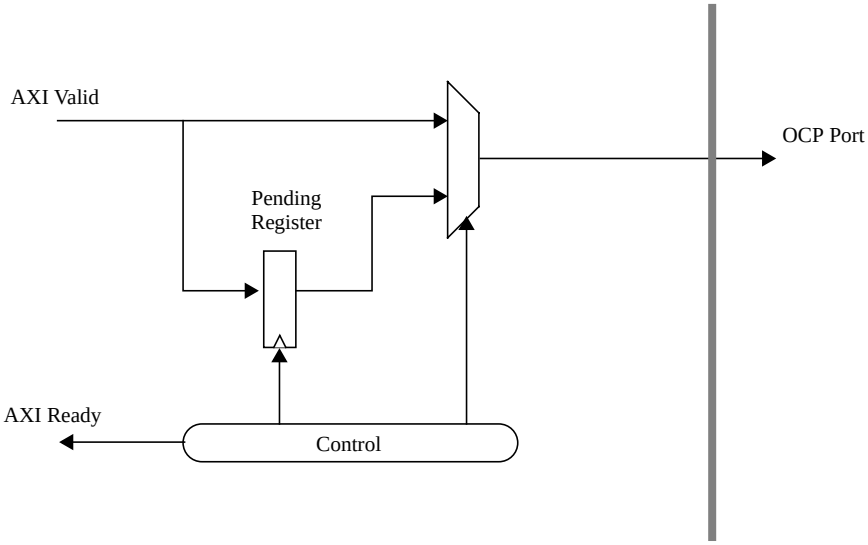


Figure 3.8 Input Port Diagram

3.4.2 AXI Command Map

The command mapping block is a combinational block that maps AXI encodings to OCP encodings. The detailed mapping are described in the following sections.

3.4.2.1 ARADDR/AWADDR Mapping

Since the OCP port data interface is from 64 to 256 bits wide, OCP addresses are 64 to 256 bit-aligned. The AXI addresses are byte addresses, and the lower address bits are encoded in *OC_MByteEn* for Reads and *OC_MDataByteEn* for Writes. Table 3.13 shows the bus configuration according to the width of the data bus.

Table 3.13 ARADDR/AWADDR Mapping

AXI Address	OC_MAddr	Bus Width
AWADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:3]*, 3'h0	64 bits
ARADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:3]*, 3'h0	64 bits
AWADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:4], 4'h0*	128 bits
ARADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:4]*, 4'h0	128 bits
AWADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:5]*, 5'h0	256 bits
ARADDR['MBB_ADDR_WIDTH-1:0]	*OC_MAddr['MBB_ADDR_WIDTH-1:5]*, 5'h0	256 bits

3.4.2.2 AWVALID/ARVALID Mapping

An ARVALID on the AXI Read address channel is translated to a OCP Read command. A configuration pin (*'MBB_A2O_CFG_WRNP*) dictates whether the bridge translates *AWVALID* on the AXI Write address channel into a WRITE/WRNP command on OCP. When the bridge is used to connect to the ScratchPad RAM interfaces, the configuration pin should be set to 0, since the slave OCP interface on the ScratchPad RAM only supports a WRITE command (it still generates WRITE responses). It should be set to 1 when connected to the IO Coherence Unit on the 1004K CPS.

Table 3.14 AWVALID/ARVALID Mapping

	'MBB_A2O_CFG_WRNP	OC_MCmd	Notes
ARVALID	-	READ	OCP Read
AWVALID	0	WRITE	OCP Write
	1	WRNP	OCP Write Non-posted

3.4.2.3 AWLEN/ARLEN

These are mapped to *OC_MBurstLength*. The *AWLEN*/*ARLEN* go from 0-15, while the *OC_MBurstLength* encoding goes from 1-16.

3.4.2.4 AWSIZE/ARSIZE

ARSIZE is expected to be 8 bytes for transactions with a burst length greater than 1. For a 64-bit wide data bus and AXI Read transactions with a burst length of 1, *ARSIZE* is translated into *OC_MByteEn* as shown below. *AWSIZE* is unused. *WSTRB* determines the data-lane enables for WRITE transfers.

Table 3.15 ARSIZE Mapping for 64-bit Wide Data Bus

ARADDR[2:0]	ARSIZE[2:0]	OC_MByteEn['MBB_BUS_DEFTYPE*8-1:0]
000	000	0000_0001
000	001	0000_0011
000	010	0000_1111
000	011	1111_1111

Table 3.15 ARSIZE Mapping for 64-bit Wide Data Bus (Continued)

ARADDR[2:0]	ARSIZE[2:0]	OC_MByteEn[‘MBB_BUS_DEFTYPE*8-1:0]
001	000	0000_0010
001	001	0000_0110
001	010	0001_1110
001	011	1111_1110
010	000	0000_0100
010	001	0000_1100
010	010	0011_1100
010	011	1111_1100
011	000	0000_1000
011	001	0001_1000
011	010	0111_1000
011	011	1111_1000
100	000	0001_0000
100	001	0011_0000
100	010	1111_0000
100	011	1111_0000
101	000	0010_0000
101	001	0110_0000
101	010	1110_0000
101	011	1110_0000
110	000	0100_0000
110	001	1100_0000
110	010	1100_0000
110	011	1100_0000
111	000	1000_0000
111	001	1000_0000
111	010	1000_0000
111	011	1000_0000

3.4.2.5 AWBURST/ARBURST

The 2 burst types supported on the AXI interface are incrementing (INCR) and wrapping (WRAP) bursts. The IO Coherence Unit in the 1004K CPS supports wrapping bursts only for burst lengths of 4, i.e., addresses that wrap at 32-byte boundaries. Wrapping bursts are not supported for other burst lengths or address wrap boundaries.

Table 3.16 ARBURST/AWBURST Mapping

AWBURST/ARBURST	OC_MBurstSeq
INCR	INCR
WRAP	WRAP

3.4.2.6 ARID/AWID/WID

ARID and *AWID* are directly mapped onto *OC_MTagID*. *WID* is directly mapped onto *OC_MDataTagID*.

3.4.2.7 ARLOCK/AWLOCK

These are not used by the bridge

3.4.2.8 ARCACHE/AWCACHE and ARPROT/AWPROT

These are not used by the bridge. Users have full flexibility to map these to *OC_MReqInfo* and *OC_MConnID* via the *ARSIDEBAND* and *AWSIDEBAND* signals. These would be needed only if the AXI-OCP bridge is connected to the IO Coherence Unit (IOCU) of the 1004K CPS.

3.4.2.9 ARSIDEBAND/AWSIDEBAND

Please refer to Section 3.4.8 “Sideband Signals” for a description of how these sideband signals are mapped.

3.4.3 OCP Port

The OCP port selects an AXI transaction from either the AXI Read Channel or the AXI Write channel and puts it on the OCP port. It also manages OCP flow control for command and write data. The arbitration between AXI Read and Write ports is done as shown in Table 3.17. The arbitration logic has to select from up to 4 commands in the AXI Read and Write address channels. There can be 1 command in the pending register and 1 command on the channel per address port.

Table 3.17 AXI Port Arbitration

ar_vld	aw_vld	ar_pend_reg_vld	aw_pend_reg_vld	port_sel
0	0	0	1	aw_pend_reg
0	0	1	0	ar_pend_reg
0	0	1	1	ar_pend_reg/aw_pend_reg
0	1	0	0	aw
0	1	0	1	aw_pend_reg

ar_vld	aw_vld	ar_pend_reg_vld	aw_pend_reg_vld	port_sel
0	1	1	0	ar_pend_reg
0	1	1	1	ar_pend_reg/aw_pend_reg
1	0	0	0	ar
1	0	0	1	aw_pend_reg
1	0	1	0	ar_pend_reg
1	0	1	1	ar_pend_reg/aw_pend_reg
1	1	0	0	ar/aw
1	1	0	1	aw_pend_reg
1	1	1	0	ar_pend_reg
1	1	1	1	ar_pend_reg/aw_pend_reg

Column 1 in Table 3.17, *ar_vld*, indicates that there is a valid command on the Read Address AXI channel in this cycle. Column 2, *aw_vld*, indicates that there is a valid command on the Write Address AXI channel in this cycle. Column 3, ar_pend_reg_vld, indicates that there is a valid command in the pending register of the AXI Read port. Column 4, aw_pend_reg_vld, indicates that there is a valid command in the pending register of the AXI Write port. Column 5 indicates which of the 4 sources (ar, aw, ar_pend_reg, aw_pend_reg) makes it to the OCP port in that cycle. The pending registers always have priority over the new commands on the AXI channels. When both ar_vld and aw_vld are set or when both ar_pend_reg_vld and aw_pend_vld_reg are set, the arbiter uses a round-robin scheme to make a selection.

Write data is transferred on the OCP bus only after the Write command is put on the OCP bus. Once a write command is put on the AXI bus, no new write commands are put on the bus till the write data phase for the completes on the OCP bus. However, new read commands can be transferred on the OCP bus while the write data phase is pending.

3.4.4 Pending Transaction FIFO

All AXI transactions require a response and are hence non-posted by definition. The AXI-OCP bridge converts AXI transactions into non-posted OCP transactions on the OCP port. Responses to transactions with the same TagID have to be received in order, but there is no ordering restriction for transactions with different TagID's. So when a response is received on the OCP interface the bridge needs to be able to route this response to either the AXI read response channel or the AXI Write Response channel. This is accomplished by storing the transaction type for outstanding OCP transactions in a per tag id transaction FIFO. There are 16 FIFOs using tagged ID to compare and select each entry in the FIFO. Each entry in the FIFO is a bit indicating whether the transaction was a Read or a Write. Each FIFO is 24 entries deep to match up with the maximum number of transactions supported by the IOCU slave interface. When an OCP response is received, the STagID is used to lookup the appropriate FIFO and determine the transaction type for the response. This is then used to return a response to the Response Control unit which, in turn, forwards it to the appropriate AXI channel.

3.4.5 Response Control

This unit behaves differently between the case where the AXI-OCP bridge is connected to an IOCU slave port (static input *OC_MRespAcceptEn* = 1) and the case where it is connected to the DMA port of one the ScratchPad RAM blocks (static input *OC_MRespAcceptEn* = 0).

In the first case the IOCU slave interface implements OCP response flow control via *OC_MRespAccept* and so OCP responses are passed straight through to the appropriate AXI response port with the *OC_MRespAccept* coming from the corresponding AXI response READY signal.

In the second case the ScratchPad DMA interface does not support *OC_MRespAccept* so the Response Unit does the following to, internally, implement the *OC_MRespAccept* function.

- Throttle the OCP command and write data channels so that only one command at a time is presented to the scratchpad DMA interface.
- Store the returning OCP responses until the READY signal becomes active on the AXI read or write response channel and the stored response can be passed on. Note that because of the command side throttling storage for only one response is required.

3.4.6 AXI-OCP Bridge Latency

The AXI-OCP bridge adds 1 cycle latency on the request path. There is no additional latency on the response.

3.4.7 AXI Slave Interface Requirements

The bridge imposes the following restrictions on an AXI master that connects to its AXI slave interface:

1. Write data interleaving is NOT supported.
2. The ISPRAM and DSPRAM OCP ports on MIPS32 cores, can only handle transactions with a burst length of 1.
3. The IOCU OCP slave port can support bursts with burst lengths of 1 to 16. But the size of the bursts has to be 64-bits. Narrow transfers are not supported for AXI bursts of length greater than 1.
4. Incrementing bursts are supported with burst lengths from 1-16. Wrap bursts are only supported for bursts of length 4 and address should wrap at a 32 byte boundary. Note that, if the bridge is connected to the ScratchPad RAM interfaces, only bursts of length 1 are supported.

3.4.8 Sideband Signals

The bridge includes sideband signals that can be used by the system designer to pass additional information from the AXI to the OCP domain. Note that these signals are not part of either protocol, but are included for maximum flexibility. This feature can be used when the AXI-OCP bridge is used to connect to the IO Coherence Unit of the 1004K CPS. The following 2 inputs to the IOCU can be mapped from the *AWSIDEBAND*/*ARSIDEBAND* for each transaction.

- *OC_MReqInfo*
- *OC_MConnID*

The width of these 2 signals is configurable in the configuration file (`mbb_config.vh`). The width of the AXI sideband signals is the sum of the widths of these two OCP signals. If these signals are not going to be used, it is recommended to connect the input sideband signals to 0.

The format of the *ARSIDEBAND*/*AWSIDEBAND* for use with the IOCU is shown in Figure 3.9 below.

Figure 3.9 ARSIDEBAND/AWSIDEBAND Format

The use of *MConnID* and *MreqInfo* for use with the IOCU are described in the 1004K CPS documentation.

Chapter 4

Clocking and Reset Methodology

This chapter describes the clocking and reset scheme used in the BusBridge™ 3 Modules and contains the following sections:

4.1 Clocking Methodology

The sequential methodology used within the BusBridge 3 Modules is a very simple synchronous design style. D-type, positive-edge triggered flip-flops are the only kind of sequential elements that are used. These flops will generally be tested with a full-scan methodology, but the exact approach depends on the capabilities of the standard cell library chosen by the implementor.

The clock generation unit resides outside of the BusBridge 3 Modules. This chapter only lays out requirements on the clock input to the modules.

4.2 Clock Domains

The BusBridge 3 Modules have a single clock input (ACLK). This is typically the System or interconnect clock. The requirements on this clock input are:

- OCP-AXI

  ACLK needs to be a supported synchronous divisor of the MIPS32 core clock. The logic to support the clock divisor on this interface is handled by the MIPS32 core. The scheme for a frequency divisor on the OCP interface is described in the Integrators Guide of the various MIPS32 cores or the Users Manual of the SOC-it L2 and the 1004K Coherent Processing System.

- AXI-OCP

  - If the AXI-OCP bridge is connected to the IOCU of the 1004K CPS, AXI clock has to be derived as a supported synchronous divisor of the CM clock. Please refer to the 1004K CPS Users Manual for details of the supported divisors.

  - If the AXI-OCP bridge is connected to DMA interface of the ScratchPad RAMs on MIPS32 cores, AXI clock has to be derived with the same supported synchronous clock divisor as on the main OCP port of the MIPS32 core. Please refer to the appropriate MIPS32 Core Integrator’s Guide for more details.

4.3 Clock Gating

The term *gated clock* is used to refer to a derived clock that is created by logically AND'ing an unconditional clock with a logic signal (called the *condition*) which suppresses the high pulse of the unconditional input clock. The wide conditional registers in the BusBridge 3 Modules can be configured to use fine grained clock gating. Clock gating can be configured by editing the configuration file located at `$MIPS_PROJECT/proc/config/customer/mbb_config.vh`. The default option is clock gating enabled (see below).

```
`define MBB_CREGW_MODULE        mvp_mbb_cregister_gc
```

If fine grained clock gating is not desired, the line above should be modified to:

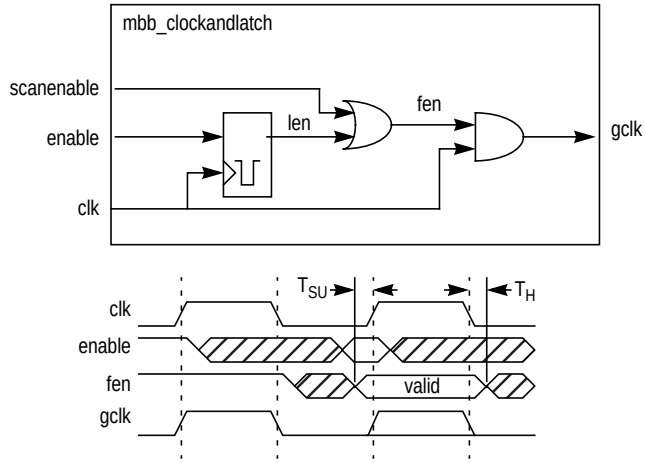
```
`define MBB_CREGW_MODULE        mvp_mbb_cregister_ngc
```

The following sections discuss the basic circuit used to achieve clock gating in the BusBridge 3 Modules.

4.3.1 Standard Clock Gating Circuit

Generating the conditional signal to be logically AND'ed with an unconditional clock to create a gated clock requires special care in a flop-based environment, to ensure that no extraneous glitches are generated when the conditional gating signal changes. The BusBridge 3 Modules employ a transparent-low latch, as shown in Figure 4.1, to properly time the conditional signal before it is AND'ed with the clock.

Figure 4.1 Typical Clock Gating Circuit



This gating circuit is coded in the RTL model, encapsulated in the module `mbb_clockandlatch`, for all uses of gated clocks in the MIPS BusBridge 3. In this module, note that there is a one-to-one correspondence between the transparent latch and the AND gate, which allows the module to be placed as a single element if desired. In this way, the physical clock connection to the latch and AND gate can be identical or at least very close. Since the *gclk* output goes to conditional flops whose clock insertion delays should be matched against all other conditional or unconditional flops in the same domain, the *clk* input will need to be supplied from an earlier tap in the clock tree.

Furthermore, note that a scan-related signal is used to force the gated clock active during scan. Refer to the Physical Design Guide for further details about the impact of gated clocks on testability.

There are setup and hold constraints on nodes *enable* and *fen* which must be checked during static timing analysis. Node *enable* has to meet the setup and hold requirements with respect to net *clk* at the transparent-low latch. And node *fen* has to meet setup/hold requirements with respect to net *clk* at the gating element. It is generally preferable for the *clk* at the latch and *clk* at the gating element to be identical, but if there is skew between these two clock nets due to placement differences between the latch and the AND gate, then that must be accounted for when performing the setup and hold checks during post-layout static timing analysis.

In general, it is desirable to minimize the delay from *clk* to *gclk* during synthesis. Net *gclk* must be matched against the unconditional clock which is used to control unconditional flops in order to minimize hold time problems within the BusBridge 3 Modules.

Many standard cell libraries contain an integrated cell that meets the functional requirements of the `mbb_clockandlatch` module. Use of an integrated cell, when available, is usually preferable since it minimizes the placement constraints discussed above. See the *MIPS32® Physical Design Guide* for more details about incorporating an integrated library cell into the synthesis environment.

4.3.2 Fine-Grain Clock Gating of Conditional Registers

All sequential elements in the BusBridge 3 RTL models are instantiations of primitive modules whose names begin with `mvp_` and are supplied in the `$MIPS_HOME/$MIPS_CORE/proc/design/rtl/global` subdirectory of the standard BusBridge 3 distribution. Some flops in the BusBridge 3 Modules are unconditionally updated, and are represented by instantiations of the `mvp_register` module; however, a significant number of flops in the design are conditionally updated, and are coded by using instantiations of the module `mvp_cregister` or similar modules.

The conditional registers can be implemented with a mux in front of an unconditional flop, as shown in Figure 4.2, however, a configuration option is available to enable more aggressive use of gated clocking when implementing these conditional registers. This method is shown in Figure 4.3. Since a large portion of the power associated with a sequential design occurs in the clock tree, local gating of the clock tree at registers which only need to be updated some of the time can significantly minimize the effective switching of the capacitance associated with the clock network. Gated clocks may also save area, since the feedback mux before every flop can be replaced by gating logic that is amortized across multiple flops on the same condition.

Figure 4.2 Conditional Register Implementation Without Gated Clocks

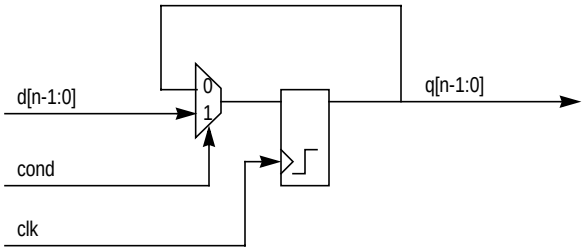
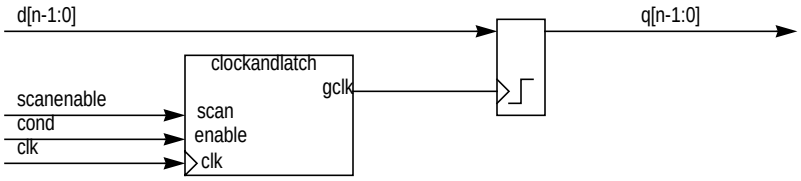


Figure 4.3 Conditional Register Implementation With Gated Clocks



All conditional registers, which are controlled by the same condition and 4 bits or more in width, are instantiated in the RTL using either the `mvp_cregister_wide` modules. These modules have a single clock and single condition input, but the width of the data input, q output, and therefore the number of conditional flops, is parameterized.

- The `mvp_cregister_wide` module is used for truly conditional registers, which must be conditionally updated regardless of whether gated clocks are desired. With no gated clocks, this module is implemented using a mux whose select is controlled by a condition signal. The mux is used to recirculate the old value of the flop when no update occurs, and to select a new value when the condition is asserted. If clock gating is desired, then the condition controls a clock gating element (via the `mbb_clockandlatch` module described previously) whose output goes to the clock input of the flops. Gated clocking, then, replaces a feedback mux for every conditional flop in the parameterized module with a single latch and AND gate. Due to the overhead of the latch/AND gate, only conditional registers with a parameterized width of four or more bits are converted to use gated clocks.

The RTL support for gated clocking of conditional registers is automatically included in the RTL and supporting scripts, but clock tree generation needs to consider the effect of additional local gated clocks during physical implementation.

4.4 Reset

4.4.1 OCP-AXI

There are 2 reset inputs to the OCP-AXI bridge.

- *OC_MResetn* is the active low reset that is generated by a master connected to the OCP interface of the bridge.
- *ARESETn* is the active low reset that is generated by the AXI domain of the SOC.

The OCP-AXI bridge is held in reset if any of these reset inputs is asserted. These reset inputs need to be driven by a Reset block. A single reset signal can drive both these reset inputs. These reset inputs should be synchronous to the clock input to the bridge (*ACLK*).

4.4.2 AXI-OCP

The AXI-OCP bridge is reset by the active low AXI reset input (*ARESETn*). The reset input should be synchronous to the clock input to the bridge (*ACLK*). This reset input is also driven out by the AXI-OCP bridge as an output reset signal *OC_MReset_n*, which is a reset signal from the OCP master port.

Chapter 5

Functional Verification

This chapter discusses the functional verification environment for BusBridge™ 3 Modules and contains the following sections:

5.1 Verification Overview

The BusBridge 3 Modules deliverables contain a testbench that supports functional verification of OCP-AXI, OCP-SPL and AXI-OCP modules. Functional verification of these modules requires running tests in a constrained random environment on the RTL model of the design in a Verilog-based simulator.

Figure 5.1 shows a block diagram of the portion of the test bench that is used to test the OCP-AXI and OCP-SPL modules. The shaded boxes are the RTL components.

If you want to simulate a single OCP-AXI, then the OCP-SPL is not needed and it should be replaced with its stub component. A stub component is just an empty shell that has the same pin-out as the component. This is accomplished by editing the “`mbb_config.vh`” file located under `$MIPS_PROJECT/proc/config/customer`. Change the define for `MBB_SPL` to the stub module as shown below.

```
`define MBB_SPL mbb_spl_stub
```

If you choose to simulate with the `OCP-SPL`, then the standalone testbench only supports the reference address decode module. This is the default in the configuration file (`mbb_config.vh`)

```
`define MBB_SPL_ADDR_DEC_MODULE mbb_spl_addr_dec
```

The rest of the blocks in the testbench are verification components. The main testbench components are:

- Transaction generator written in System Verilog (`sv_gen.sv`)

  The Transaction generator is a System verilog module that generates a random mix of Read/Write transactions based on a transaction mix read from a template file.

- OCP master model (`ocpmaster.v`)

The OCP master block converts the transactions generated by the transaction generator into transactions on the OCP bus. It is modeling the OCP bus behavior of the MIPS32 core, the 1004K Coherent Processing System, or the SOC-it L2 cache.

- Synopsys DesignWare AMBA VIP components

  With the Synopsys DesignWare AMBA Verification IP (VIP) interconnect, slave and monitor models are used to model a typical bus interconnect that the bridge would connect to in an SOC. The monitor model is used to facilitate AXI protocol checking.

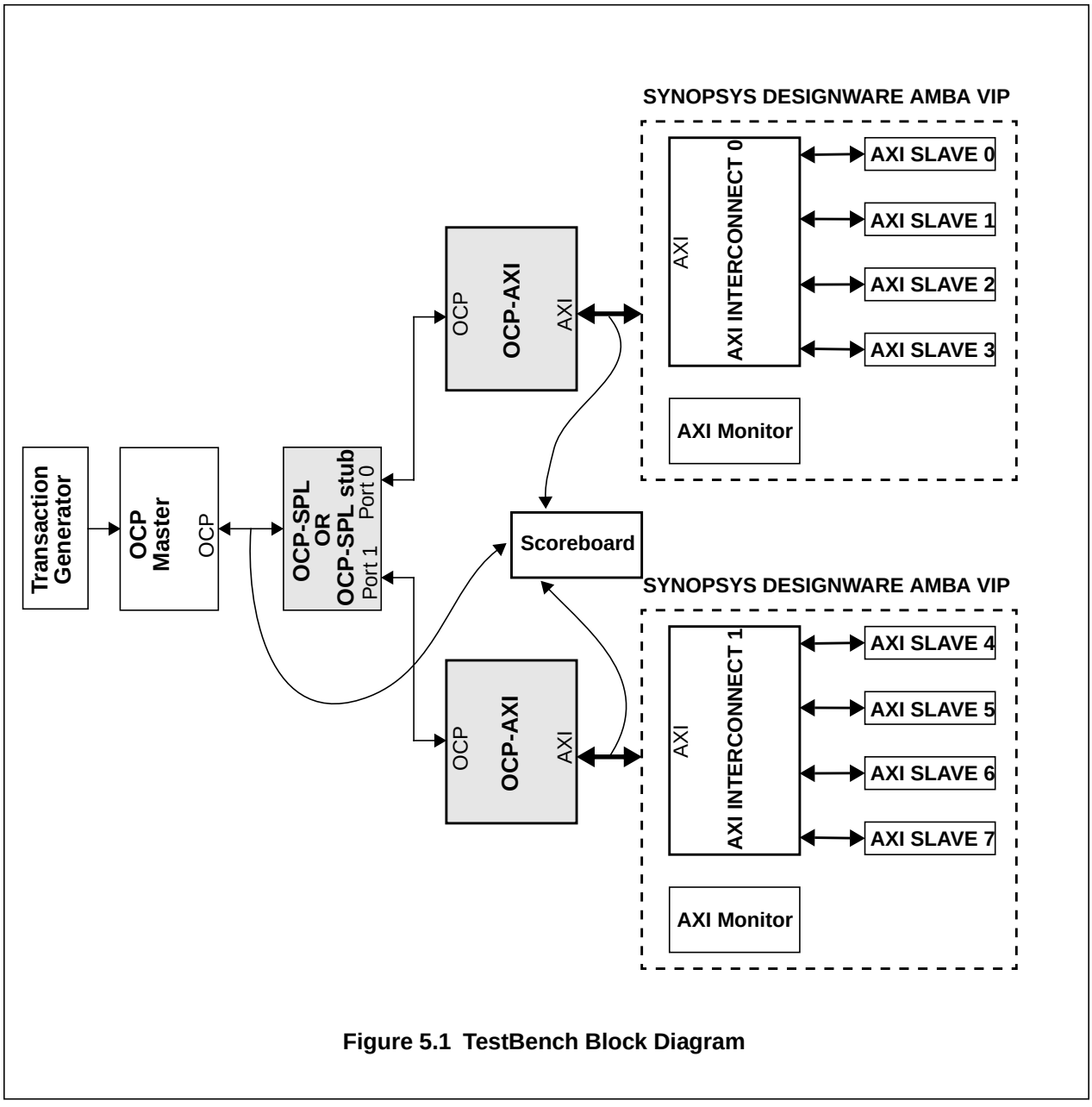


Figure 5.1 TestBench Block Diagram

The simulation reads in a template file that specifies the relative weights of transaction types. The OCP transaction generator reads in these weights and uses them to guide the distribution of transaction types and destinations. The OCP master receives the generated transactions and drives them onto the OCP bus of either the OCP-SPL or OCP-AXI.

As it drives transactions onto the OCP bus, the OCP master keeps track of a reference memory image, which is updated with the latest data values from all the writes to valid addresses. Whenever a read response returns, the OCP master checks each data beat against the expected value as tracked in the reference memory image.

In addition to checking the data of read responses, the test bench also performs AXI protocol checks. This is achieved by instantiating an AXI monitor on the AXI bus of OCP-AXI. The Scoreboard tracks transactions on the OCP interface and matches them up against transactions on the AXI interfaces. In addition, it performs other ordering checks.

Figure 5.2 shows the block diagram of the portion of the testbench that is used to validate the AXI-OCP module.

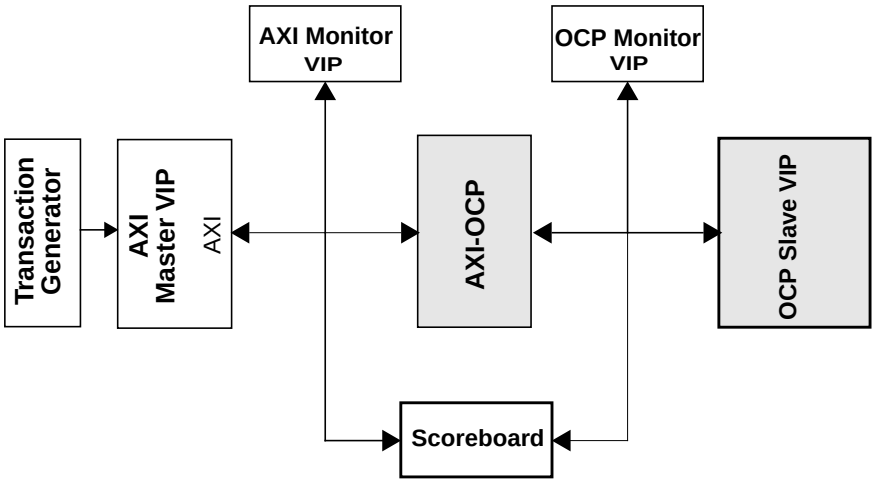


Figure 5.2 AXI-OCP Test Bench Block Diagram

The main components of this testbench are:

- Transaction generator written in System Verilog (`axi_gen.sv`)

  The Transaction generator is a System verilog module that generates a random mix of Read/Write transactions based on a transaction mix read from a template file.

- Synopsys DesignWare VIP blocks

  Synopsys DesignWare AMBA and OCP Verification IP (VIP) are used to model an AXI master and an OCP slave. AXI and OCP protocol monitors are used to enable protocol checks.

- Scoreboard

A scoreboard tracks transactions/responses on the AXI and OCP interfaces and checks the transaction mapping from AXI to OCP and response mapping from OCP to AXI.

5.2 Running Simulations

This section discusses how to run tests with a Verilog simulator. Running tests requires setting up the simulation environment, creating the simulation executable, and running the tests. Each of these steps is described below.

5.2.1 General Simulation Setup

Make sure the BusBridge 3 Modules environment has been set up as discussed in Section 2.2 “Installing a Release”. Here are some general environment issues to check:

- Verify that the `MIPS_PROJECT`,`MIPS_HOME`, and `MIPS_CORE` environment variables are set, and that the `$MIPS_HOME/$MIPS_CORE/bin`, `$MIPS_HOME/$MIPS_CORE/flow/bin`, and `$MIPS_HOME/$MIPS_CORE/flow/verif/bin` directories are in your path.

5.2.2 Simulation Setup for Synopsys VCS

To simulate with VCS, set your `MIPS_SIM_TYPE` environment variable:

```
% setenv MIPS_SIM_TYPE vcs
```

In order to run a simulation using VCS, several things must first be set up:

- The setup is dependent on the local network environment, but at a minimum, the VCS executables must be in the path and the `VCS_HOME` environment variable must be set.

5.2.3 Creating the Simulation Executable

The `$MIPS_HOME/$MIPS_CORE/bin/buildSim` script is used to create the simulation executable of the core in the user’s specific environment.

Before making the executable, make sure the `MIPS_SIM_TYPE` environment variable has been set:

```
% echo $MIPS_SIM_TYPE
```

If it is not set, see the previous section for your specific simulator.

The simulation executable must be built in an empty directory and on the local machine:

```
% cd $MIPS_PROJECT

% cd <build_dir>

% buildSim -b sa_rtl [-d build_dir] -local -define [OCP2AXI | AXI2OCP]
```

A description of the command line options follows:

- `-b sa_rtl`: This simulates the RTL with the standalone testbench described in this chapter.

- `-d build_dir`: The simulator executable will be built in the directory `build_dir`. The default build directory is the current directory.
- `-local`: Perform the build on the local machine.
- `-define`: Choose the design being simulated. The choices are:
  - `OCP2AXI` (for OCP-AXI, OCP-SPL and OCP-AXI2)
  - `AXI2OCP` (for AXI-OCP)

The following sequence must be used to properly clean up an existing build directory, before performing a new build.

```
% cd <build_dir>
% rm -rf *
```

5.2.4 Running Random Tests

The `bin` subdirectory includes the Perl `runrandoms` script for functional verification of the BusBridge 3 Modules. This script and the file generated at run-time to conveniently rerun the test are described in the following sections.

- Section 5.2.4.1 “runrandoms Script”
- Section 5.2.4.2 “rerun Command File”

Results of the tests are saved in subdirectories under the current directory. Each test directory name starts with `mps_random_<random seed>`

5.2.4.1 runrandoms Script

This script simulates one, or more, random tests. It must be invoked in an empty directory:

```
% cd $MIPS_PROJECT

% mkdir <sim_dir>

% cd <sim_dir>
```

To simulate OCP-AXI or OCP-AXI2, `runrandoms` is used as shown below

```
% runrandoms [-o “+dual_axi +debug_axi“] [-t <number of transactions>] <run
count> <simulation model path> <template name> [-w]
```

`runrandoms` runs as many simulations as specified by `<run count>`. It executes the simulator executable in the directory given by the absolute path `<simulation model path>`. The simulator executable must already exist before `runrandoms` can be invoked. A build script is used to create the desired simulator executable and is explained in Section 5.2.3 “Creating the Simulation Executable”. The template file specified by `<template name>` is read in by the simulation testbench to set the relative weights of the types and destinations for the random transactions. In the example below, the `runrandoms` command specifies running 10 random tests using the default template. The number of transactions in each test is not specified and defaults to 5000.

The `-o` string controls the following simulation behavior:

- `+dual_axi` : Simulates the Dual AXI bridge model with the OCP-SPL (`mbb_spl` instantiated). The default is just a single AXI bridge (OCP-AXI) where the OCP-SPL is replaced with its stub component (`mbb_spl_stub`).

- `+debug_axi` : Enables AXI monitor transaction logging.

- `-w` : Creates a wavefoem dump of the simulation.

In the example below, the `runrandoms` command specifies running 10 random tests using the default template. The number of transactions in each test is not specified and defaults to 5000. Since no -o string is specified, it simulates a single AXI bridge and no AXI logging enabled.

```
% runrandoms 10 $MIPS_PROJECT/build_dir/sim default
```

The next example specifies running 20 random tests, each having 10000 transactions. It simulates the Dual Axi model (OCP-AXI2). AXI monitor logging is also enabled.

```
% runrandoms -o “+dual_axi +debug_axi” -t 10000 20 /home/me/build_dir/sim
default
```

The `runrandoms` script performs the following steps:

1. Creates a new directory `mbb_random_<random seed>` for each test under the current directory.

2. Copies the specified template file from `$MIPS_HOME/$MIPS_CORE/proc/verification/templates` into the working directory.

3. Prepare other necessary files and links for running the simulation.

4. Execute the simulation and produce the log file. If the model was built with dump enabled, the dump file containing waveforms of internal signals is also produced.

To simulate AXI-OCP, runrandoms is used as shown below

```
% runrandoms [-o “+debug_axi“] [-t <number of transactions>] <run count>
<simulation model path> <template name> [-w]
```

The `log` file details the template weights used to generate the test, the simulation output, and the self-checking test status. The end of the `log` file lists the summary results for the test along with statistics on transaction count and latencies for each slave that was accessed. The test could end with 3 results:

- Test PASSED - This implies that the test ran sucessfully and passed.

- Test FAILED - This could be a due to a read response check failure or a protocol failure or a failure reported by the scoreboard.

- Test PASSED (with WARNINGS)- This category includes the following

    - Warnings reported by the AXI/OCP verification models.

Any result other than ‘Test PASSED’ needs to be investigated and debugged.

All simulation test results are summarized in the file `result_summary`. This file lists the tests as PASSED or FAILED.

5.2.4.2 rerun Command File

It may be useful to rerun a test. The command which was invoked to run the simulation is saved in the file `rerun`, and is located in the test directory. When rerunning a test, it may be helpful to save the output from the second simulation by piping it to a file that is different from the original log file. Rerunning the test can be conveniently performed by sourcing the `rerun` file:

```
% source rerun >& log2
```

Alternatively, the command in the `rerun` file can be cut and pasted onto the Linux command line.

5.3 Debugging Simulation Runs

To debug a simulation, a waveform dump file can be created by using the [`-w`] switch with `runrandoms` as described in Section 5.2.4.1 “runrandoms Script”x.

Various `log/trace` files are created during a simulation run, which contain useful information for debug. This section describes the files that are generated in a simulation run directory.

5.3.1 Result Files

The results for each simulation run in stored in a subdirectory under the current directory when `runrandoms` was invoked.

The following files are created when you run a simulation for OCP-AXI or OCP-AXI2.

- `template`: The specified template file is copied and renamed to `template`, so the testbench always reads in the file with this exact name. This file specifies the relative weights of the choices made while randomly generating the transactions.
- `log`: Output from the simulation run. The result of the simulation is shown at the end of this file. This log also contains a transaction information from the AXI monitor on the AXI interface of the bridge.
- `axi_monitor_0.log`: This log contains transaction information from the AXI monitor on the AXI interface of the bridge. AXI monitor logging is only enabled if the test is run with the `+debug_axi` argument.
- `axi_monitor_1.log`: This log contains transaction information from the AXI monitor on the AXI interface of the bridge connected on Port 1 of the OCP-SPL. This file is created only if the test is run with the `+dual_axi` mode. i.e., with the OCP-SPL. AXI monitor logging is only enabled if the test is run with the `+debug_axi` argument.
- `master_ocp.dump`: Transaction Trace on the OCP bus of the OCP2AXI bridge.
- `Scoreboard_0.log:` Transaction trace on OCP and Port 0 AXI interfaces as seen by the scorebaord.
- `Scoreboard_1.log:` Transaction trace on OCP and Port 1 AXI interfaces as seen by the scoreboard. This file is created only if the test is run with the `+dual_axi` mode, i.e., with the OCP-SPL.

The following files are created when simulating AXI-OCP:

- `template`: The specified template file is copied and renamed to `template`, so the testbench always reads in the file with this exact name. This file specifies the relative weights of the choices made while randomly generating the transactions.

- `log`: Output from the simulation run. The result of the simulation is shown at the end of this file. This log also contains transaction information from the AXI monitor on the AXI interface of the bridge.

- `mbb_axi2ocp_rtl.conf`: This is the configuration file for the OCP master interface on the AXI-OCP module.

- `axi_master_monitor.log`: This log contains transaction information from the AXI monitor on the AXI interface of the bridge connected on Port 1 of the OCP-SPL. This file is created only if the test is run with the `+dual_axi` mode i.e with the OCP-SPL. AXI monitor logging is only enabled if the test is run with the `+debug_axi` argument.

- `Scoreboard_a2o.log`: Transaction trace on OCP and Port 0 AXI interfaces as seen by the scorebaord.

If a failure occurs, review the `log` file to determine the type of failure.

5.4 Creating New Templates

It is possible to develop new templates and use them to guide the random transaction generator in the MIPS test bench. New template files should be created in the `$MIPS_HOME/$MIPS_CORE/proc/verification /templates` directory.

The `default` template in that directory can be copied and edited to create a new template. The new template should have the same format and labels as the `default` template.

5.4.1 Template Files

The template is a simple text file. Each line of valid data consists of a label followed by one or more spaces followed by a non-negative integer specifying the weight for this label. Each label belongs to a category. The category is the first part of the label string until an underscore character is encountered. The weight specifies the relative frequency of the transaction type or destination indicated by its label relative to all other labels in the same category. For example, the following lines specify the weights for labels in the category `TYPE`. They specify that the `LOAD` transaction type should be given a 10/20 (50%) probability, the `STORE` transaction type be given a 7/20 (35%) probability, the `IDLE` transaction type be given a 1/20 (5%) probability, and the `SYNC` transaction type be given a 2/20 (10%) probability:

```
TYPE_LOAD   10

TYPE_STORE  7

TYPE_IDLE   1

TYPE_SYNC   2
```

Comments preceded by "//" and blank lines are allowed in the template file. All labels described below must be specified or the simulation will issue an error because the template is incomplete. To disable a label, assign it a weight of 0.

The MBB3 deliverables templates directory has 3 basic templates:

- `default` - Simulates the OCP-AXI and the OCP-AXI2 module.
- `axi_default` - Simulates the AXI-OCP module when it is connected to the IO Coherence Unit of the 1004K™ CPS.
- `axi_spram` - Simulates the AXI-OCP module when it is connected to the ScratchPad RAM modules on MIPS32 cores.

Table 5.1 OCP-AXI Template

`TYPE_LOAD`	Read transaction.
`TYPE_STORE`	Write transaction.
`TYPE_IDLE`	No transaction (idle bus).
`TYPE_SYNC`	SYNC transaction. A SYNC transaction is visible to the bridge when it is externalized by a MIPS32 core or SOC-it L2 or the coherence manager of a MIPS32 1004K coherent processing system.
`BURSTSIZE_8`	Transaction is a burst of 8 OCP data beats. This is only possible with a SOC-it L2 configured with a 64-byte line size.
`BURSTSIZE_4`	Transaction is a burst of 4 OCP data beats.
`BURSTSIZE_1`	Transaction is a single OCP read or write.
`RESPACCEPT_1`	These 2 labels are used to randomize the *OC_MRespAccept* input to the bridge. These are relative weights the control the value of *OC_MRespAccept* over the period of the simulation run. *RESPACCEPT_1* should never be 0.
`RESPACCEPT_0`	
`READY_1`	These 2 labels are used to modify the default value of the AXI ready signals at the interconnects master and slave interfaces. If *READY_1* is set to 1, then the default values of the AXI ready at the interconnect master and slave ports is 1. If *READY_0* is 1, then the testbench randomly picks default values and delays for the AXI ready signals. If both of these labels are set to 1 in the template, the testbench randomly picks one of them.
`READY_0`	

Table 5.2 Template for AXI-OCP Template

`TYPE_RD`	Read transaction.
`TYPE_WR`	Write transaction.

TYPE_SINGLE	Transactions with a burst length of 1
TYPE_BURST	Transactions with a burst length from 2 to 16. This is set to 0 in the axi_spram template, since the ScratchPad modules in the MIPS32 cores only support single transactions.
TYPE_MAXTAGS	Maximum number of Tags used by the AXI Master.
READY_1	These 2 labels are used to modify the default value of the AXI ready signals generated by the AXI master. If *READY_1* is set to 1, then the default values of “ready” generated by the AXI master is 1. If *READY_0* is 1, then the testbench randomly picks default values and delays for the AXI ready signals. If both of these labels are set to 1 in the template, the testbench randomly picks one of them.
READY_0	
TYPE_MBB_A2O_CFG_WRNP_1	If this is set to 1, the testbench sets the ‘*MBB_A2O_CFG_WRNP* pin to 1, which configures the bridge to generate WRNP commands for OCP Write transaction. This is used to model the behavior of the bridge when it is connected to the IOCU of the MIPS32 1004K CPS.
TYPE_MBB_A2O_CFG_WRNP_0	If this is set to 1, the testbench sets the ‘*MBB_A2O_CFG_WRNP* pin to 0, which configures the bridge to generate WR commands for OCP Write transaction. This is used to model the behavior of the bridge when it is connected to the DMA interfaces of the ScratchPad RAMs on MIPS32 cores. If both labels are set to 1, the testbench randomly picks one of these.
TYPE_MRESPACCEPT_EN	If this is set to 0, the testbench sets *theOC_MRespAcceptEn* pin to 0, which models the behavior of the bridge when it is connected to the ScratchPad RAMs on MIPS32 cores. If this is set to 1, the *OC_MRespAcceptEn* is set to 1 and it models the behavior of the bridge when it is connected to the IOCU interface of the MIPS32 1004K CPS.

Chapter 6

Waveforms

This chapter describes the waveforms of the BusBridge™ 3 Modules' transactions and contains the following sections:

- Section 6.1 "OCP-AXI Bridge Waveforms"
- Section 6.2 "AXI-OCP Bridge Waveforms"

6.1 OCP-AXI Bridge Waveforms

This waveforms in this section illustrate the conversion by the OCP-AXI bridge of the OCP transactions generated by MIPS32 CPU cores into AXI transactions. In the waveform figures, note that the clock is the AXI system clock. The MIPS32 CPU cores generate three types of transactions: Read, Write, and Sync. The figures below describe Read and Write transactions. A Sync transaction is a single OCP Read to a specific address, so the waveforms for Sync will be the same as the waveforms for a Read.

6.1.1 Single Read Command

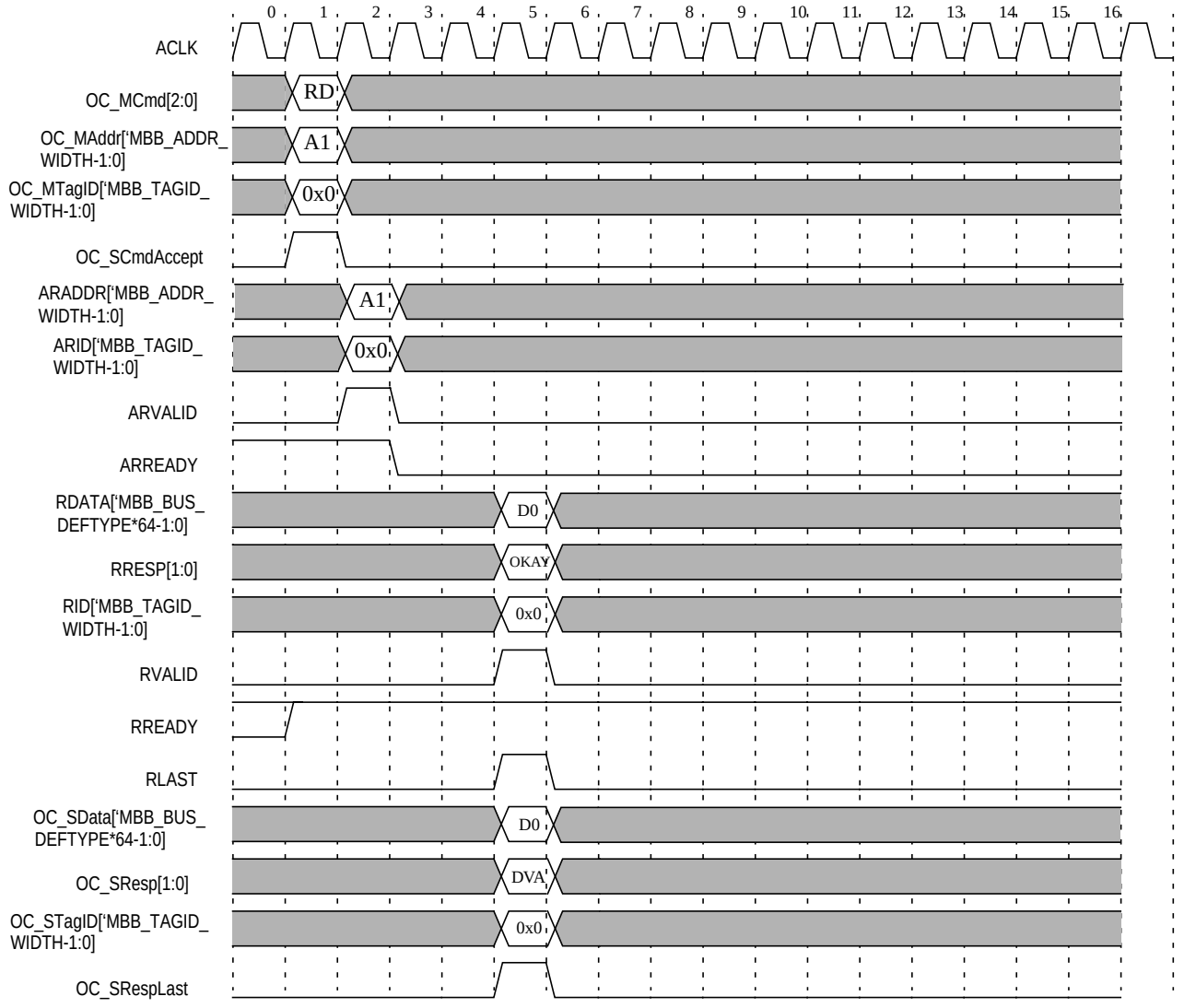


Figure 6.1 OCP Single Read Translated to AXI Single Read

Figure 6.1 shows a single OCP Read command that gets translated into a single AXI Read command. The figure shows the important signals from the command and data phases of the OCP bus and AXI bus. There are other signals on both interfaces that are not shown here. There is a 1 cycle latency on the request phase, but no additional latency on the response. There are no wait states in this example. The OCP command is accepted by the bridge in cycle 1 and *ARVALID* is asserted on the AXI interface in cycle 2. Since *ARREADY* is high, the AXI Read command is accepted by the AXI subsystem of the SOC in cycle 2. The AXI subsystem returns response data in cycle 5. The bridge always drives *RREADY* to 1, meaning that it will always accept read response data. The read response data is returned to the OCP interface in cycle 5 with no additional latency.

6.1.2 Burst Read Command

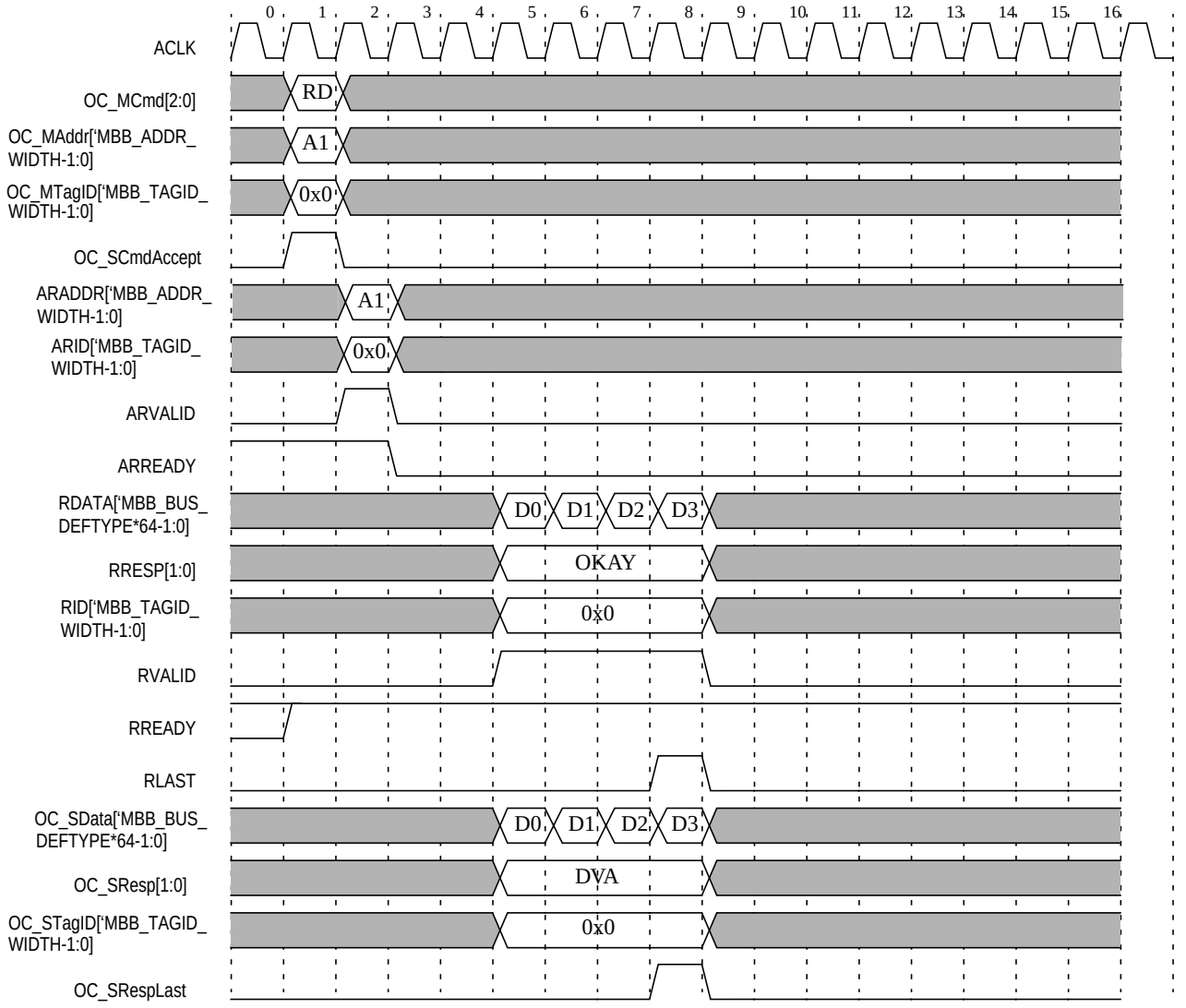


Figure 6.2 OCP Burst Read Translated to AXI Burst Read

Figure 6.2 shows a burst OCP read command that gets translated into a burst AXI Read command.

6.1.3 Read Command with Wait States

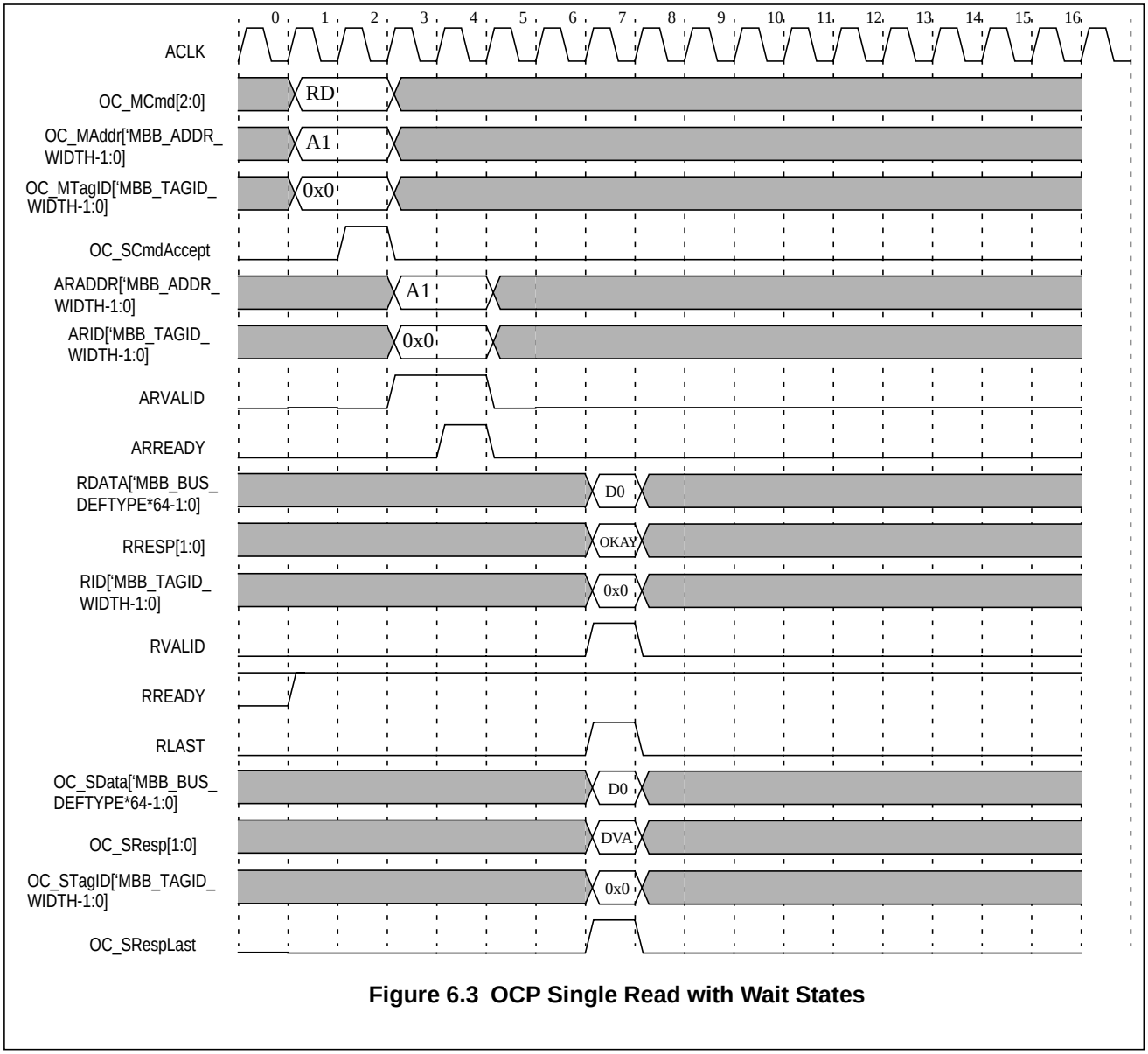


Figure 6.3 OCP Single Read with Wait States

Figure 6.3 shows a OCP read command in cycle 1, which cannot be accepted by the bridge in cycle 1. The bridge de-asserts *OC_SCmdAccept* in cycle 1 to hold off the OCP read command. The command is accepted by the bridge in cycle 2, by asserting *OC_SCmdAccept* and it is driven onto the AXI interface in cycle 3. This example also illustrates a case where the AXI system has *ARREADY* de-asserted so the AXI read command has to be held by the bridge for an additional cycle when the AXI subsystem asserts *ARREADY* and accepts the read command in cycle 4.

6.1.4 Single Write Command

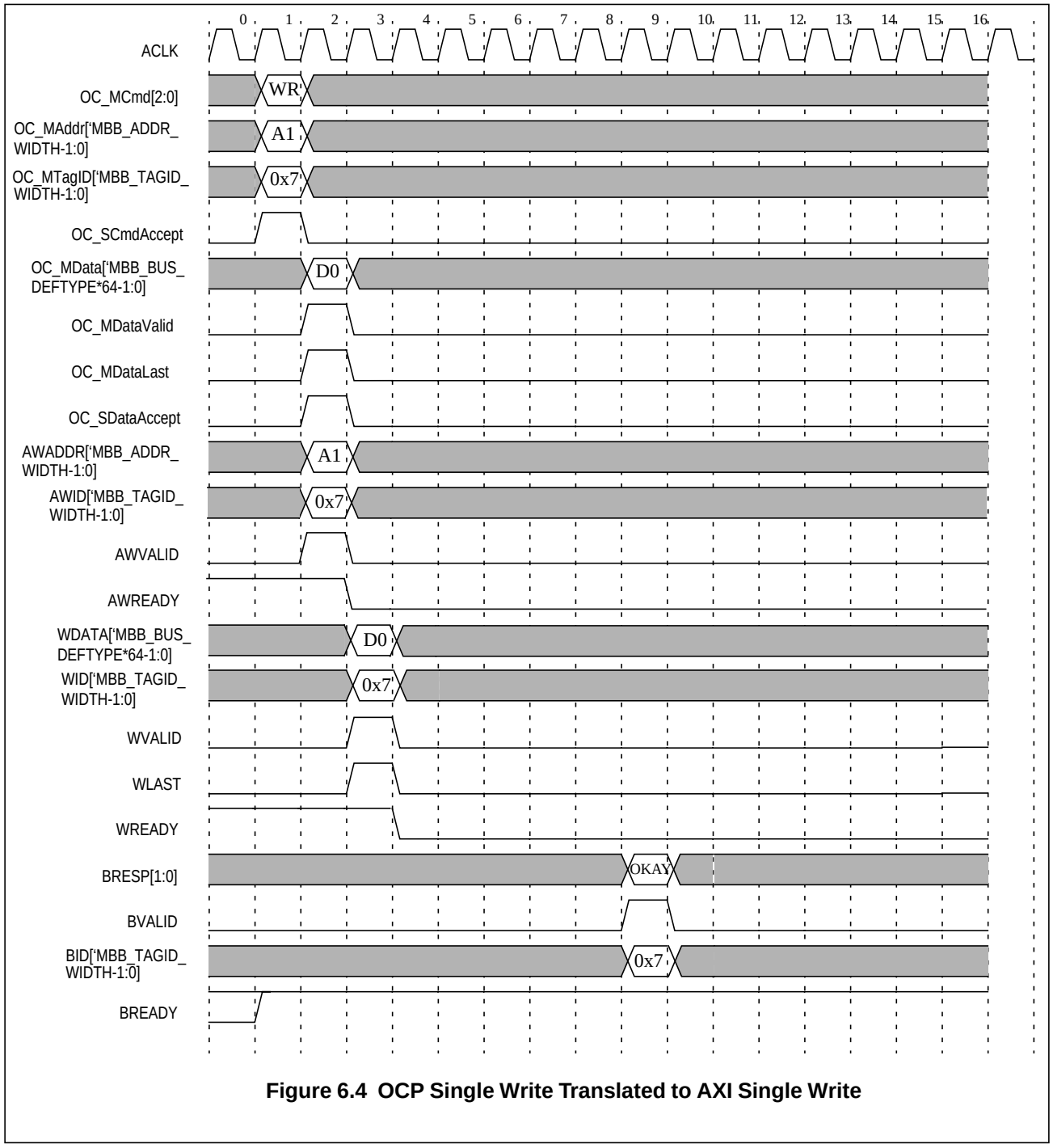


Figure 6.4 OCP Single Write Translated to AXI Single Write

Figure 6.4 depicts a single OCP Write command being translated into a single AXI write command. The OCP write command is driven on the AXI address and data channels in cycles 2 and 3. An AXI write response for the write transaction is received in cycle 9. The bridge always has *BREADY* asserted, thus completing the write transaction.

6.1.5 Burst Write Command

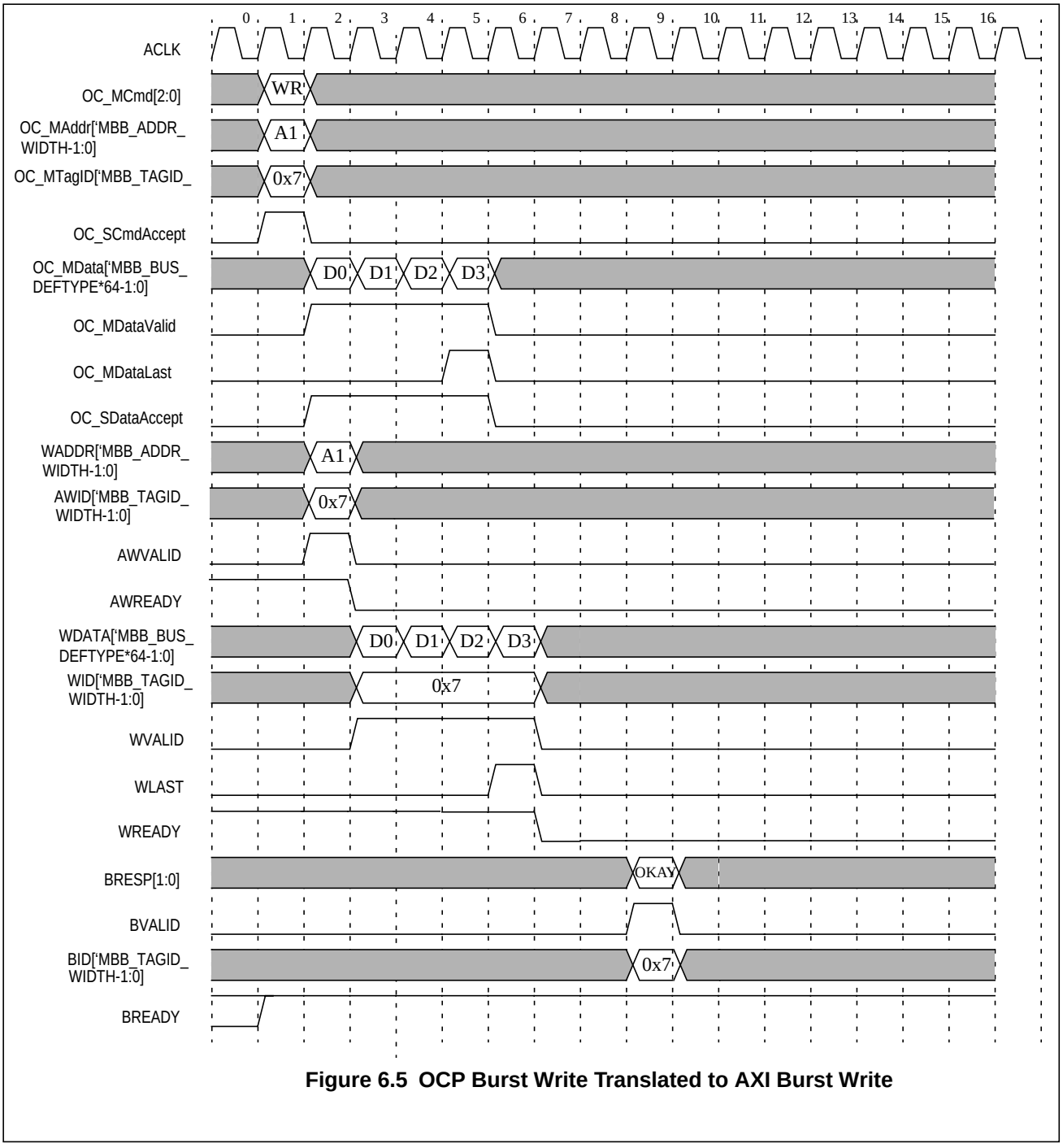


Figure 6.5 OCP Burst Write Translated to AXI Burst Write

Figure 6.5 depicts a burst OCP Write command being translated into a burst AXI write command.

6.1.6 Write Command with Wait States

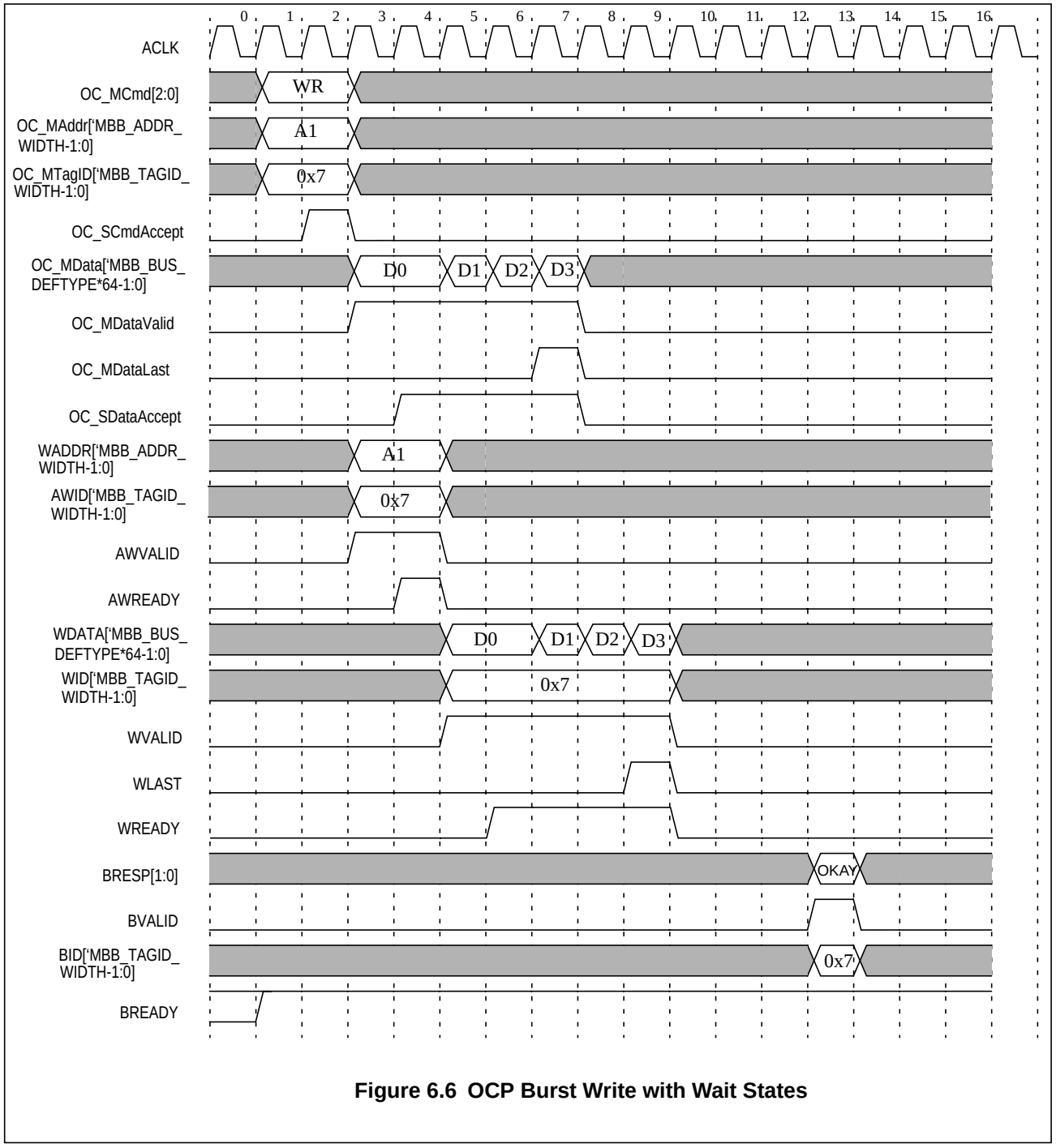


Figure 6.6 OCP Burst Write with Wait States

Figure 6.6 depicts a OCP write command that is flow controlled by the bridge in cycle 1. It is accepted by the bridge in cycle 2 by de-asserting the *OC_SCmdAccept*. This example also illustrates the case where the SOC's AXI sub-system needs an additional cycle to accept the AXI write command on the AXI write address channel. This is because

the default value of *AWREADY* is 0. It transitions to 1 in cycle 4 following the assertion of *AWVALID*. This scenario is also illustrated on *WREADY* of the write data channel.

6.2 AXI-OCP Bridge Waveforms

This waveforms in this section illustrate the conversion of OCP transactions generated by MIPS32 CPU cores into AXI transactions, by the OCP-AXI bridge. In the waveform figures, note that the clock is the AXI system clock.

6.2.1 Single Read

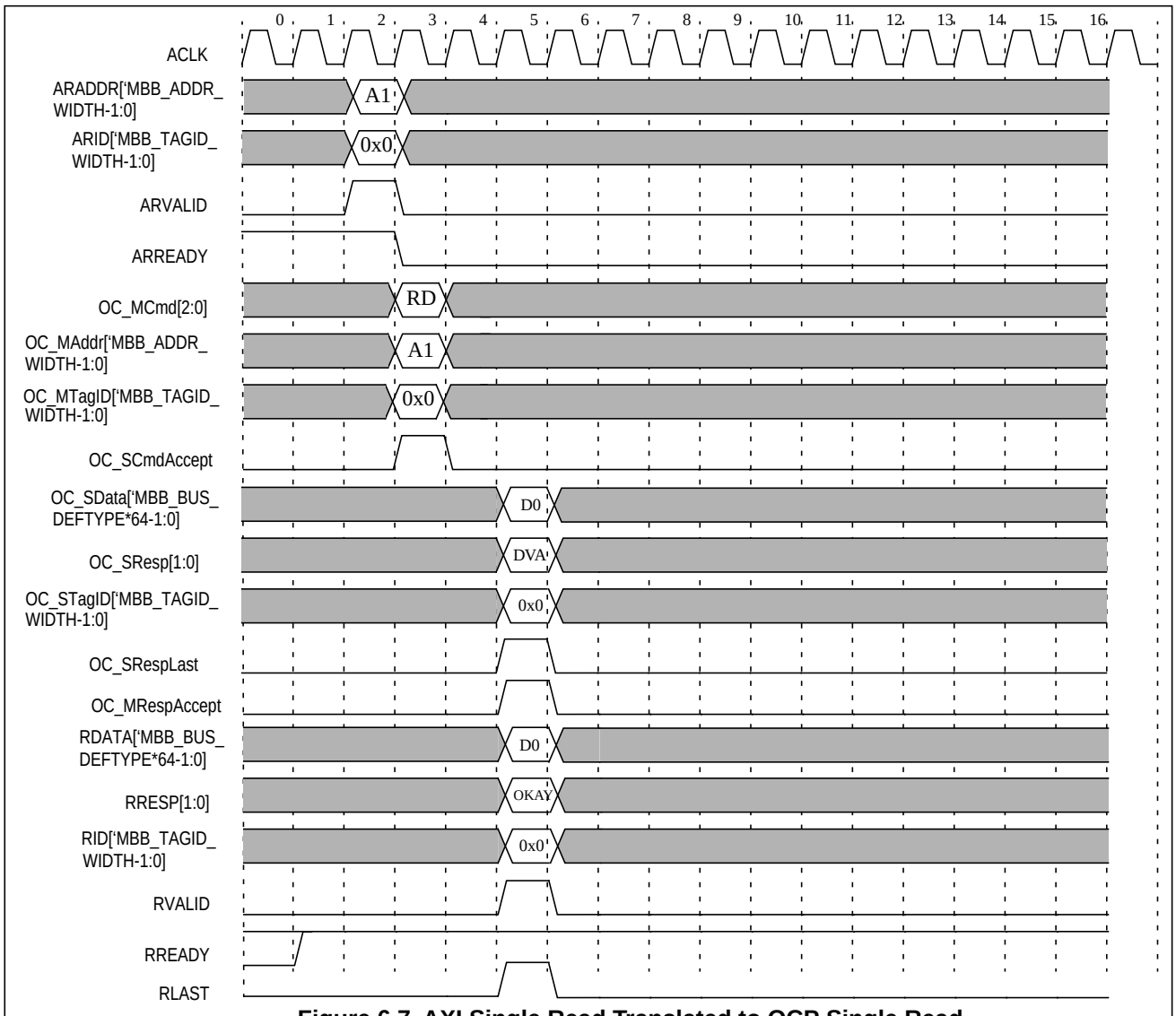


Figure 6.7 AXI Single Read Translated to OCP Single Read

Figure 6.7 shows a single Read command on AXI converted to a single read command on OCP.

6.2.2 Burst Read with AXI Master Wait States

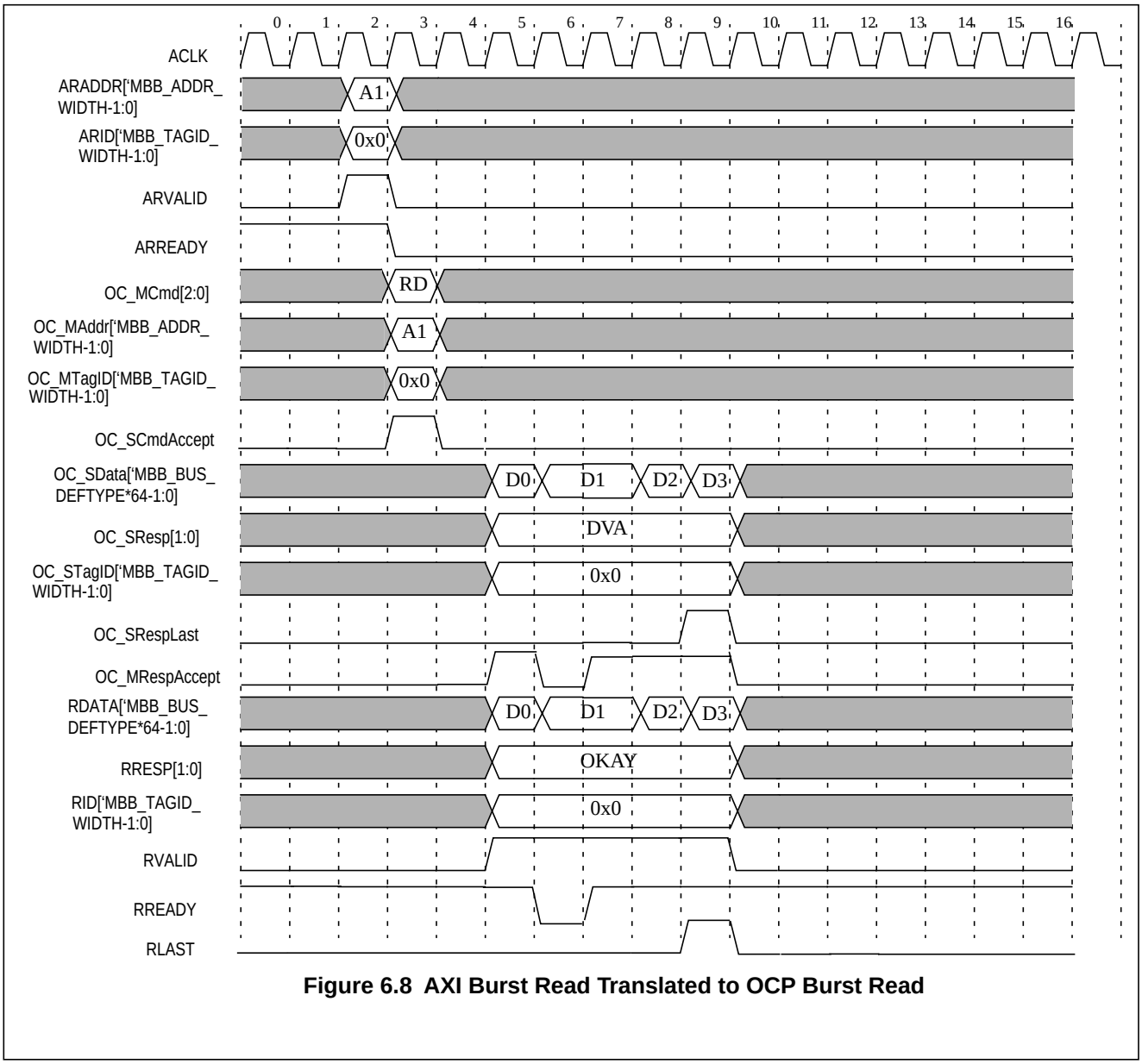


Figure 6.8 AXI Burst Read Translated to OCP Burst Read

Figure 6.8 shows and AXI burst read translated by the bridge into an OCP burst Read. It also shows wait states inserted by the AXI master on the response. This wait state translates into a OCP wait state inserted by the OCP master interface of the AXI-OCP bridge, via the signal *OC_MRespAccept*.

6.2.3 Single Write Command

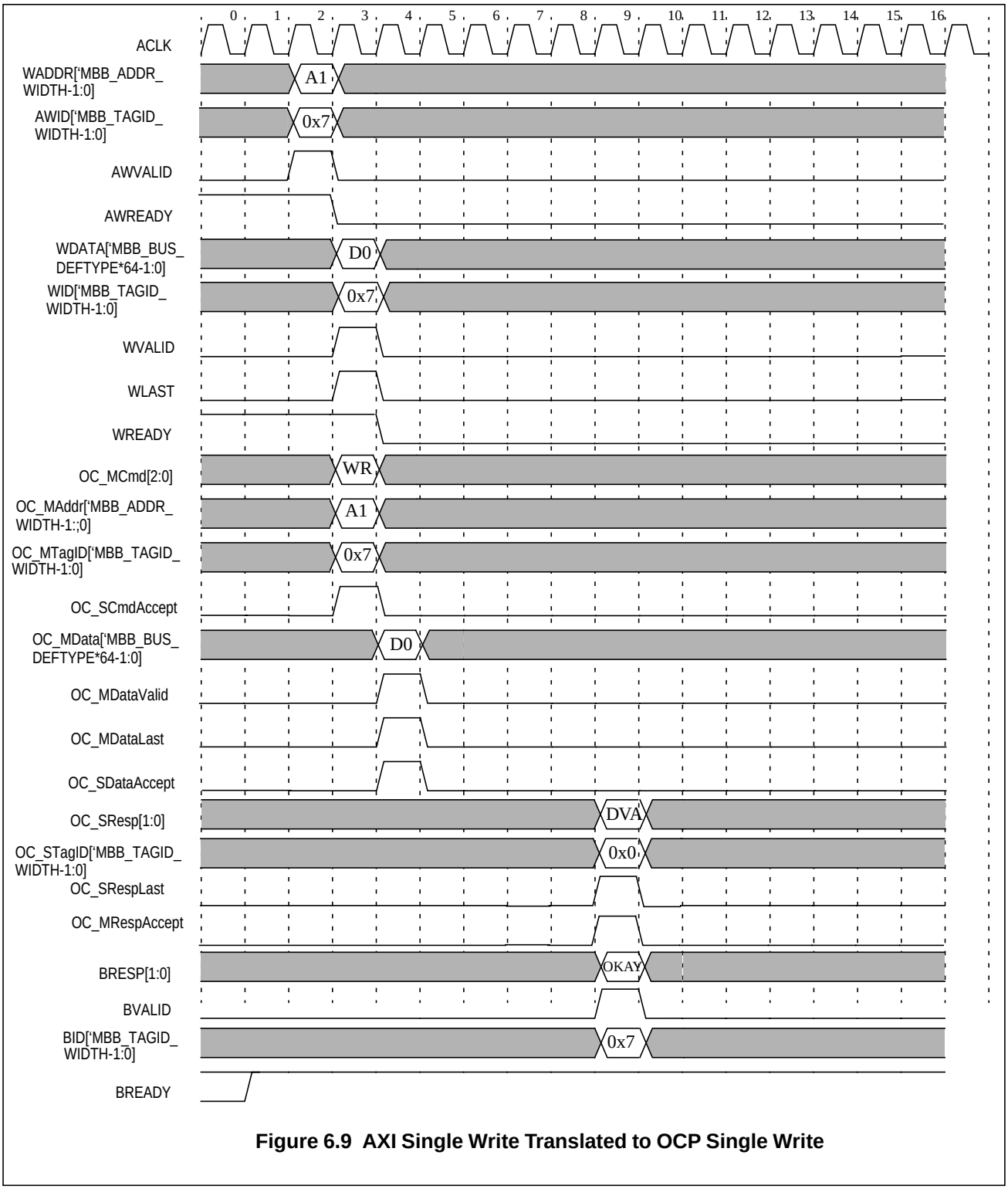


Figure 6.9 AXI Single Write Translated to OCP Single Write

6.2.4 Burst Write with Wait States

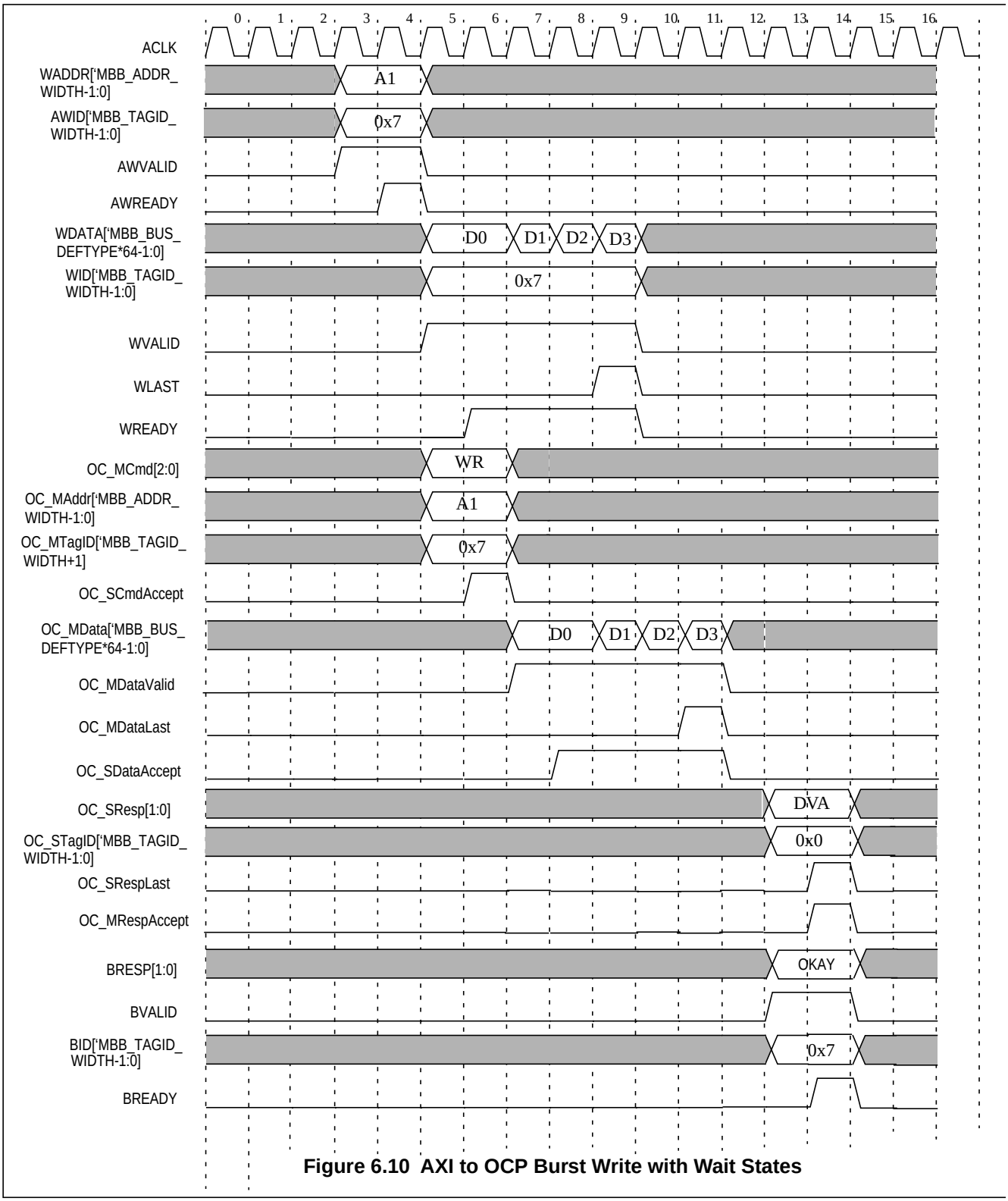


Figure 6.10 AXI to OCP Burst Write with Wait States

Figure 6.10 shows an AXI burst Write with Wait States on the Write address channel and on the Write Data channel. This example also shows that the AXI master inserts a WAIT state on the Write response channel, which manifests itself as a wait state inserted by the OCP master via *OC_MRespAccept*. It should be noted that Write Data is allowed to precede the Write command on the AXI interface. The AXI-OCP bridge will delay the write data till after the write command has been accepted on the OCP interface.

Chapter 7

Synthesis

This chapter describes the synthesis methodology, the reference flow, and synthesis steps for the BusBridge™ 3 Modules. It contains the following sections:

- Section 7.1 “Methodology”
- Section 7.2 “Flow File Structure”
- Section 7.3 “Synthesis”

7.1 Methodology

A Design Compiler (DC) flow is supported for the topdown synthesis of the MIPS BusBridge 3 modules. The reference flow consists of the DC scripts and constraints.

7.2 Flow File Structure

All the flow scripts required for synthesizing the MIPS BusBridge 3 modules are contained in the `$MIPS_PROJECT/flow/synth/cosyn` and `$MIPS_PROJECT/flow/synth/projsyn` directories. This `projsyn` directory contains following sub-directories that need to be modified by the user:

- `lib`: This directory contains the library-specific setup files for DC, for running synthesis. These need to be modified by the user to set library paths, constraints, etc.

7.3 Synthesis

7.3.1 Preparing for Synthesis

The MIPS BusBridge 3 Modules configuration file (`$MIPS_PROJECT/proc/config/customer/mbb_config.vh`), must be edited to choose whether fine- grained clock gating is used. Fine-grained clock gating is enabled by default in the configuration file. If fine-grained clock gating is not desired, then the configuration file should be edited as shown below:

```
‘define MBB_CREGW_MODULE        mvp_mbb_cregister_ngc
```

For synthesis, three links are required to define the location of the library information files, the modules described with gate-level implementation, and configuration information. These three links must have specific names and should be created as described below:

```
% cd $MIPS_PROJECT/proc
% mkdir build_<name_of_build>
% cd build_<name_of_build>
% ln -s ../../flow/synth/projsyn/lib/<library_vendor_specific_dir> techlib
```

```
% ln -s ../design/gate/<library_vendor_specific_dir> gatemodules
% ln -s ../config/<configuration_name> config
```

A sample of all the files in techlib is provided in `$MIPS_PROJECT/flow/synth/projsyn/lib/generic`

The constraints for all I/Os are listed in Chapter 8, and are coded in the file `<module_name>.iodelays.portcons`.

These constraints can be modified by the user.

7.3.2 Running Synthesis

Synthesis can be run on the following modules using the command below.

- OCP-AXI (*mbb_ocp2axi*)
- OCP-SPL (*mbb_spl*)
- OCP-AXI2 (*mbb_ocp2axi_dual*)
- AXI-OCP (*mbb_axi2ocp*)

```
% cd <build_directory>

% $MIPS_PROJECT/flow/synth/cosyn/common/syncore.pl -f topdown -t
[dc|dcxg|dcxg-topo] -d [mbb_ocp2axi | mbb_spl | mbb_ocp2axi_dual | mbb_axi2ocp] -c
-scan -compile_ultra -debug
```

After the synthesis run completes, the following directories are created:

- `log`: Contains log of DC run. The log files should be carefully reviewed for errors.
- `report`: Contains various reports generated during synthesis.
- `mapped`: Contains synthesized netlist and sdc constraint file.
- `port.rpt`: Contains constraints set on the design. These should be reviewed to make sure that the constraints are set correctly according to the user's design requirements.

Chapter 8

Port Definitions

This chapter describes the OCP-AXI signals for the AXI bridge. This chapter contains the following sections:

- Section 8.1 “Naming Conventions”
- Section 8.2 “OCP-AXI Detailed Signal Descriptions”
- Section 8.3 “OCP-SPL Detailed Signal Descriptions”
- Section 8.4 “OCP-AXI2 Detailed Signal Descriptions”
- Section 8.5 “AXI-OCP Detailed Signal Descriptions”

8.1 Naming Conventions

8.1.1 Signal Direction

The direction key for the signal descriptions is shown in Table 8.1. These symbols are used in the column labeled “Dir” in Table.

Table 8.1 AXI Bridge Signal Direction Key

Dir	Description
I	Input of the AXI bridge, sampled on the rising edge of the appropriate clock signal
O	Output of the AXI bridge, unless otherwise noted, driven at the rising edge of the appropriate clock signal.
SI	Static inputs to the AXI bridge. These signals are tied to either power or ground and should not change state when reset is deasserted.
SO	Static output from the AXI bridge.

8.2 OCP-AXI Detailed Signal Descriptions

Table 8.2 lists the pinout of the AXI bridge. The OCP and AXI interfaces are not fully registered. However, all signals are synchronous to the rising edge of the primary AXI clock, *ACLK*. Outputs on the interface may have an amount of logic after the preceding flop(s), and inputs may go through some combinational logic before being registered by the bridge. Other signals pass though some combinatorial logic in the bridge before going out to output ports.

The expression of timing constraints for the AXI bridge depends on many factors, such as maximum target frequency, process technology, standard cell library characteristics, etc., so it is difficult to provide a generic set of tim-

ing guidelines that will apply in all situations. The “Timing” column in Table 8.2 shows the timing of the AXI bridge interface signals, expressed as a percentage of the minimum target period. All the input and output directions are with respect to the AXI bridge. For an output, the timing number means percentage of the cycle after the driving clock edge when the data is available. For an input, the number means percentage of the cycle from the preceding clock edge when data is required.

Table 8.2 OCP-AXI Bridge Signals

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td colspan="4">OCP Signals</td></tr>
<tr><td>OC_MCmd[2:0]</td><td>I</td><td>OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on OC_MReqInfo and OC_MAddrSpace. The encodings used by the OCP master are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Used for data write and L2 CACHE write or invalidate</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Used for fetch or data read or L2 CACHE reads or SYNC.</td></tr>
<tr><td>3-7</td><td>Unused</td><td>-</td><td>Not used</td></tr>
</table></td><td>75%</td></tr>
<tr><td>OC_MReqInfo[6:0]</td><td>I</td><td>OCP command bus extension.
For transactions other than SYNC and CACHE, the OC_MReqInfo[2:0] field encodes the cacheability attributes for a transaction.
OC_MReqInfo[3] indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.
The encoding of the OC_MReqInfo field for all transactions other than CACHE is summarized in the following table:
Encoding for all Transactions Other Than CACHE
<table>
<tr><th>Encoding</th><th>Command Information</th></tr>
<tr><td>0</td><td>Cacheable, noncoherent, WT, NWA</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>2</td><td>Uncached</td></tr>
<tr><td>3</td><td>Cacheable, noncoherent, WB, WA</td></tr>
<tr><td>4-6</td><td>Reserved</td></tr>
<tr><td>7</td><td>Uncached accelerated</td></tr>
<tr><td>8-9</td><td>Reserved</td></tr>
<tr><td>10</td><td>SYNC with uncached CCA</td></tr>
<tr><td>11-15</td><td>Reserved</td></tr>
</table>
The AXI bridge does not support L2/L3 commands, so only the above encodings are recognized. OC_MReqInfo[6:4] field is used for L2 cache exclusivity control, and as such are ignored by the AXI bridge.</td><td>50%</td></tr>
</table>

Table 8.2 OCP-AXI Bridge Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MAddrSpace[1:0]</td><td>I</td><td>L2/L3 Address Space indicator. When the OCP master is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.
Note: The AXI bridge does not support L2/L3 commands.
The encoding of this field is summarized in the following table:
<table><tr><th>Encoding</th><th>Address Space</th></tr><tr><td>0</td><td>Normal address space</td></tr><tr><td>1</td><td>L2 address space</td></tr><tr><td>2</td><td>L3 address space</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td><td>50%</td></tr>
<tr><td>OC_MAddr['MBB_ADDR_WIDTH-1:0]</td><td>I</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (OC_MByteEn) or write (OC_MDataByteEn) byte enable fields.</td><td>70%</td></tr>
<tr><td>OC_MBurstSeq[2:0]</td><td>I</td><td>Indicates type of burst sequence. The OCP master can only generate two possible values, determined by the SI_SBlock static input, as shown in the following table:
<table><tr><th>Encoding</th><th>Burst Sequence</th></tr><tr><td>2</td><td>Sequential: Critical dword first, with linear wrapping for subsequent beats</td></tr><tr><td>4</td><td>Sub-block Critical dword first, with increment/decrement for subsequent beats</td></tr><tr><td>0-1,3,5-7</td><td>Unused</td></tr></table>**Note**: The AXI bridge does **not** support sub-block ordering. SI_SBlock must be hardwired to 0.</td><td>Unused</td></tr>
<tr><td>OC_MTagID[`MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Transaction tag identifier.
Note: All writes have tag ID of 4‘h7</td><td>60%</td></tr>
<tr><td>OC_MBurstPrecise</td><td>SI</td><td>Indicates whether the burst length is precise. Burst lengths are always fixed at 4 beats (or 8 beats based on configuration), so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstSingleReq</td><td>SI</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstLength[2:0]</td><td>I</td><td>Number of 64b data transfers.
<table><tr><th>Encoding</th><th>Number of Transfers</th></tr><tr><td>1</td><td>1, single transfer</td></tr><tr><td>4</td><td>4-beat burst</td></tr><tr><td>8</td><td>8-beat burst</td></tr><tr><td>others</td><td>Unused</td></tr></table></td><td>45%</td></tr>
</table>

Table 8.2 OCP-AXI Bridge Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the OC_MByteEn field to the returned read data bytes is shown in the following table:
<table>
<tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>65%</td></tr>
<tr><td>OC_MCmdSideBand['MBB_SIDEBAND_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MCmd. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification, and is used to pass user-defined signals to the AXI domain (see ARSIDEBAND and AWIDEBAND).</td><td>35%</td></tr>
<tr><td>OC_MConnID[`MBB_O2A_CONNID_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MCmd. The size of the signal is configurableby setting the value of ‘MBB_O2A_CONNID_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification and is used to pass user-defined signals to the AXI domain (see ARSIDEBAND and AWIDEBAND).</td><td>35%</td></tr>
<tr><td>OC_MDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Write data bus from the OCP master</td><td>35%</td></tr>
<tr><td>OC_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the OC_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use OC_MByteEn for transferring byte enables.
<table>
<tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>35%</td></tr>
</table>

Table 8.2 OCP-AXI Bridge Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MDataValid</td><td>I</td><td>Valid write data on OC_MData bus.</td><td>25%</td></tr>
<tr><td>OC_MDataTagID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Write data tag identifier (for out of order returns). All writes by the OCP master should have a TagID value of 0x7.</td><td>10%</td></tr>
<tr><td>OC_MDataLast</td><td>I</td><td>Last data in a write burst - only valid when OC_MDataValid is asserted.</td><td>10%</td></tr>
<tr><td>OC_MDataSideBand['MBB_SIDEBAND_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MData. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification, and is used to pass user defined signals to the AXI domain (see WSIDEBAND).</td><td>10%</td></tr>
<tr><td>OC_MReset_n</td><td>I</td><td>Active low indication that the core is being reset and other OCP devices should be reset as well.</td><td>30%</td></tr>
<tr><td>OC_SDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Returned read data to core.</td><td>55%</td></tr>
<tr><td>OC_STagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Return transaction tag ID. See OC_MTagID for encoding.</td><td>55%</td></tr>
<tr><td>OC_SResp[1:0]</td><td>O</td><td>Valid response from system controller. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td>Should not be used by OCP master</td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals bus error exception</td></tr>
</table></td><td>60%</td></tr>
<tr><td>OC_SRespInfo[1:0]</td><td>SO</td><td>OC_SRespInfo[0] indicates the type of error reported on OC_SResp. Valid only when OC_SResp shows a response error (value of 0x3).
<table>
<tr><th>OC_SRespInfo[0]</th><th>Description</th></tr>
<tr><td>0</td><td>Bus Error</td></tr>
<tr><td>1</td><td>Cache Error</td></tr>
</table>
OC_SRespInfo[1] indicates the cache line state of return data. L1 D-cache will install the line as the state that is specified on this field.
<table>
<tr><th>OC_SRespInfo[1]</th><th>Description</th></tr>
<tr><td>0</td><td>Clean line return</td></tr>
<tr><td>1</td><td>Modified/Dirty line return.</td></tr>
</table>
Note: This signal is statically driven to 2‘b00</td><td>NA</td></tr>
<tr><td>OC_SRespLast</td><td>O</td><td>Marks last data in read burst.</td><td>55%</td></tr>
<tr><td>OC_SCmdAccept</td><td>O</td><td>AXI bridge notifies the OCP master that the command is accepted.</td><td>50%</td></tr>
<tr><td>OC_SDataAccept</td><td>O</td><td>AXI bridge notifies the OCP master that the write data is accepted.</td><td>15%</td></tr>
<tr><td>OC_MRespAccept</td><td>I</td><td>OCP master accepts the read response. Note that this signal should be hardwired to 1‘b1 if the bridge is connected to an OCP master that does not implement response flow control.</td><td>55%</td></tr>
<tr><td colspan="4">AXI Signals</td></tr>
<tr><td colspan="4">Global Signals</td></tr>
</table>

Table 8.2 OCP-AXI Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
ACLK	I	Global clock signal. All signals are sampled at the rising edge of the global clock.	NA
ARESETn	I	Global reset signal. This signal is active LOW.	30%
Write Address Channel			
AWID['MBB_TAGID_WIDTH-1:0]	O	Write address ID. This signal is the identification tag for the write address group of signals.	15%
AWADDR['MBB_ADDR_WIDTH-1:0]	O	Write address. The write address bus gives the address of the first transfer in a write burst transaction.	15%
AWLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst.	15%
AWSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst.	15%
AWBURST[1:0]	O	Burst type	15%
AWLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer.	15%
AWCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.	15%
AWPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.	15%
AWVALID	O	Write address valid. This signal indicates that valid write address and control information is available.	15%
AWSIDEBAND['MBB_O2A_CONNIDSIDEBAND_WIDTH:0]	O	Sideband signal associated with the write command. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH and 'MBB_O2A_CONNID_WIDTH defines in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
AWREADY	I	Write address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Write Data Channel			
WID['MBB_TAGID_WIDTH-1:0]	O	Write ID tag. this signal is the ID tag of the write data transfer.	15%
*WDATA['MBB_BUS_DEFTYPE*64-1:0]*	O	Write data	15%
*WSTRB['MBB_BUS_DEFTYPE*8-1:0]*	O	Write strobes. This signal indicates which byte lanes to update in memory.	15%
WLAST	O	Write last. This signal indicates the last transfer in a write burst.	15%
WVALID	O	Write valid. This signal indicates that valid write data and strobes are available.	15%
WSIDEBAND['MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the write data. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (See *OC_MDataSideBand*).	15%
WREADY	I	Write ready. This signal indicates that the slave can accept the write data.	30%
Write Response Channel			
BID['MBB_TAGID_WIDTH-1:0]	I	Response ID. The Identification Tag of the write response.	Unused

Table 8.2 OCP-AXI Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
BRESP[1:0]	I	Write response. This signal indicates the status of the write transaction.	25%
BVALID	I	Write response valid. This signal indicates that a valid write response is available.	25%
BREADY	SO	Response ready. This signal indicates that the AXI bridge can accept the response information. Note that the AXI bridge drives this signal statically to a value of 1.	NA
Read Address Channel			
ARID['MBB_TAGID_ WIDTH-1:0]	O	Read address ID. This signal is the identification tag for the Read address group of signals.	15%
ARADDR['MBB_ADDR_ WIDTH-1:0]	O	Read address. The Read address bus gives the address of the first transfer in a Read burst transaction.	15%
ARLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst.	15%
ARSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst.	15%
ARBURST[1:0]	O	Burst type	15%
ARLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer.	15%
ARCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.	15%
ARPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.	15%
ARVALID	O	Read address valid. This signal indicates that valid Read address and control information is available.	15%
ARSIDEBAND'[MBB_ O2A_CONNIDSIDEBAND _WIDTH:0]	O	Sideband signal associated with the read command. The size of the signal is configurable by setting the value of '`MBB_SIDEBAND_WIDTH` and '`MBB_O2A_CONNID_WIDTH` defines in the configuration file (`mbb_config.vh`). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
ARREADY	I	Read address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Read Data Channel			
RID['MBB_TAGID_ WIDTH-1:0]	I	Read ID tag. This signal is the ID tag of the read data group of signals.	55%
*RDATA['MBB_BUS_ DEFTYPE*64-1:0]*	I	Read data	55%
RRESP[1:0]	I	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.	60%
RLAST	I	Read last. This signal indicates the last transfer in a read burst.	55%
RVALID	I	Read valid. This signal indicates that the required read data is available and the read transfer can complete.	60%
RREADY	O	Read ready. This signal indicates that the master can accept the read data and response information.	55%
Miscellaneous			
SCANENABLE	I	Scan enable signal. Used to enable conditional registers during scan operation.	30%
SCANMODE	I	Scan mode signal	30%

Table 8.2 OCP-AXI Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
SCANIN[x:0]	I	Scan-in chains. The number of scan chains is determined by the value `MBB_NUM_SCAN_CHAIN` in the config file.	30%
SCANOUT[x:0]	I	Scan-out chains. The number of scan chains is determined by the value `MBB_NUM_SCAN_CHAIN` in the config file.	30%
AXI_WERR_INT	O	Write error interrupt signal. This signal is asserted when the AXI bridge decodes an error write response.	15%
AXI_WERR_ADDR['MBB_ADDR_WIDTH-1:0]	O	Write error address signal. This signal has the doubleword address of the write that caused the error response.	15%
AXI_WERR_TYPE	O	Write error type signal. This signal encodes the error type (0 = slave error, 1 = address decode error).	15%
AXI_WERR_ACK	I	Write error interrupt acknowledge. This signal is used to de-assert the *AXI_WERR_INT* signal.	25%
AXI_CMD_PROT[1:0]	I	Command protection bits. These bits map directly to the privileged/secure bits in *ARPROT[1:0]* and *AWPROT[1:0]*. They are included since there is no equivalent signals on the OCP side. The user may hardwire the signal to some default value, or include some external logic to generate them based on the OCP command info. Note that the signal associates with the current command on the OCP bus.	20%
SI_SBlock	SO	This port indicates the burst ordering mode supported by the bridge. This signal is statically driven to 0, indicating that only sequential ordering is supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . L2 OCP Master: Connect to port `SI_L2_SBlock` of L2 . Core OCP Master: Connect to port `SI_SBlock` of core . CM OCP Master: Connect to port `SI_SBlock` of CM	NA
SI_SimpleBE	SO	This port indicates byte enables support by the bridge. This signal is statically driven to 1, indicating that only simple byte enables are supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . Core OCP Master: Connect to port `SI_SimpleBE` of core . CM OCP Master: Connect to port `SI_SimpleBE` of CM	NA
SI_SyncTxEn	SO	This port indicates support for externalized SYNC command. This signal is statically driven to 1, indicating that the bridge supports external SYNC command. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . L2 OCP Master: Connect to port `L2_SyncTxEn` of L2 . Core OCP Master: Connect to port `SI_SyncTxEn` of core . CM OCP Master: Connect to port `SI_CM_SyncTxEn` of CM	NA
SI_256b_Dp_En	SO	This port indicates support for 256 bits data mode. This signal is statically driven to 0, indicating that the bridge supports only 64 bits. This signal is used only when the CM is the OCP master. In that case, the port should be connected to port `SI_CM_256b_Dp_En` of the CM.	NA

8.3 OCP-SPL Detailed Signal Descriptions

The OCP Splitter is not a standard design. Consult the MIPS softcore product datasheet to determine whether the OCP Splitter is included in the release. If applicable, this section describes the signals used in OCP Splitter.

Table 8.3 lists the pinout of the OCP-SPL. The OCP interfaces are not fully registered. However, all signals are synchronous to the rising edge of the primary AXI clock, *ACLK*. Outputs on the interface may have an amount of logic after the preceding flop(s), and inputs may go through some combinational logic before being registered by the bridge. Other signals pass though some combinational logic in the bridge before going out to output ports.

The expression of timing constraints for the OCP-SPL depends on many factors, such as maximum target frequency, process technology, standard cell library characteristics, etc., so it is difficult to provide a generic set of timing guidelines that will apply in all situations. The “Timing” column in Table 8.3 shows the timing of the OCP-SPL interface signals, expressed as a percentage of the minimum target period. All the input and output directions are with respect to the OCP-SPL. For an output, the timing number means percentage of the cycle after the driving clock edge when the data is available. For an input, the number means percentage of the cycle from the preceding clock edge when data is required.

Table 8.3 OCP-SPL Signals

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td colspan="4">OCP Source Port Signals</td></tr>
<tr><td>OC_MCmd[2:0]</td><td>I</td><td>OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on OC_MReqInfo and OC_MAddrSpace. The encodings used by the OCP master are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Used for data write and L2 CACHE write or invalidate</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Used for fetch or data read or L2 CACHE reads or SYNC.</td></tr>
<tr><td>3-7</td><td>Unused</td><td>-</td><td>Not used</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MReqInfo[6:0]</td><td>I</td><td>OCP command bus extension.
For transactions other than SYNC and CACHE, the OC_MReqInfo[2:0] field encodes the cacheability attributes for a transaction.
OC_MReqInfo[3] indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.
The encoding of the OC_MReqInfo field for all transactions other than CACHE is summarized in the following table:
Encoding for all Transactions Other Than CACHE<table><tr><th>Encoding</th><th>Command Information</th></tr><tr><td>0</td><td>Cacheable, noncoherent, WT, NWA</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>Uncached</td></tr><tr><td>3</td><td>Cacheable, noncoherent, WB, WA</td></tr><tr><td>4-6</td><td>Reserved</td></tr><tr><td>7</td><td>Uncached accelerated</td></tr><tr><td>8-9</td><td>Reserved</td></tr><tr><td>10</td><td>SYNC with uncached CCA</td></tr><tr><td>11-15</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>OC_MAddrSpace[1:0]</td><td>I</td><td>L2/L3 Address Space indicator. When the OCP master is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.

The encoding of this field is summarized in the following table:<table><tr><th>Encoding</th><th>Address Space</th></tr><tr><td>0</td><td>Normal address space</td></tr><tr><td>1</td><td>L2 address space</td></tr><tr><td>2</td><td>L3 address space</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>OC_MAddr[‘MBB_ADDR_WIDTH-1:0]</td><td>I</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (OC_MByteEn) or write (OC_MDataByteEn) byte enable fields.</td><td>60%</td></tr>
<tr><td>OC_MBurstSeq[2:0]</td><td>I</td><td>Indicates type of burst sequence. The OCP master can only generate two possible values, determined by the SI_SBlock static input, as shown in the following table:<table><tr><th>Encoding</th><th>Burst Sequence</th></tr><tr><td>2</td><td>Sequential: Critical dword first, with linear wrapping for subsequent beats</td></tr><tr><td>4</td><td>Sub-block Critical dword first, with increment/decrement for subsequent beats</td></tr><tr><td>0-1,3,5-7</td><td>Unused</td></tr></table></td><td>Unused</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MTagID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Transaction tag identifier.
Note: All writes have tag ID of 4'h7</td><td>60%</td></tr>
<tr><td>OC_MBurstPrecise</td><td>SI</td><td>Indicates whether the burst length is precise. Burst lengths are always fixed at 4 beats (or 8 beats based on configuration), so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstSingleReq</td><td>SI</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstLength[2:0]</td><td>I</td><td>Number of 64b data transfers.
<table>
<tr><th>Encoding</th><th>Number of Transfers</th></tr>
<tr><td>1</td><td>1, single transfer</td></tr>
<tr><td>4</td><td>4-beat burst</td></tr>
<tr><td>8</td><td>8-beat burst</td></tr>
<tr><td>others</td><td>Unused</td></tr>
</table></td><td>60%</td></tr>
<tr><td>OC_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the OC_MByteEn field to the returned read data bytes is shown in the following table:
<table>
<tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>60%</td></tr>
<tr><td>OC_MDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Write data bus from the OCP master</td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the OC_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use OC_MByteEn for transferring byte enables.
<table>
<tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>60%</td></tr>
<tr><td>OC_MDataValid</td><td>I</td><td>Valid write data on OC_MData bus.</td><td>60%</td></tr>
<tr><td>OC_MDataTagID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Write data tag identifier (for out of order returns). All writes by the OCP master should have a TagID value of 0x7.</td><td>60%</td></tr>
<tr><td>OC_MDataLast</td><td>I</td><td>Last data in a write burst - only valid when OC_MDataValid is asserted.</td><td>60%</td></tr>
<tr><td>OC_SDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Returned read data to core.</td><td>60%</td></tr>
<tr><td>OC_STagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Return transaction tag ID. See OC_MTagID for encoding.</td><td>60%</td></tr>
<tr><td>OC_SResp[1:0]</td><td>O</td><td>Valid response from system controller. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td>Should not be used by OCP master</td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals bus error exception</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_SRespInfo[1:0]</td><td>O</td><td>OC_SRespInfo[0] indicates the type of error reported on OC_SResp. Valid only when OC_SResp shows a response error (value of 0x3).
<table>
<tr><th>OC_SRespInfo[0]</th><th>Description</th></tr>
<tr><td>0</td><td>Bus Error</td></tr>
<tr><td>1</td><td>Cache Error</td></tr>
</table>
OC_SRespInfo[1] indicates the cache line state of return data. L1 D-cache will install the line as the state that is specified on this field.
<table>
<tr><th>OC_SRespInfo[1]</th><th>Description</th></tr>
<tr><td>0</td><td>Clean line return</td></tr>
<tr><td>1</td><td>Modified/Dirty line return.</td></tr>
</table>
Note: This signal is statically driven to 2‘b00</td><td>60%</td></tr>
<tr><td>OC_SRespLast</td><td>O</td><td>Marks last data in read burst.</td><td>60%</td></tr>
<tr><td>OC_SCmdAccept</td><td>O</td><td>OCP-SPL notifies the OCP master that the command is accepted.</td><td>65%</td></tr>
<tr><td>OC_SDataAccept</td><td>O</td><td>OCP-SPL notifies the OCP master that the write data is accepted.</td><td>65%</td></tr>
<tr><td>OC_MRespAccept</td><td>I</td><td>OCP master accepts the read response. Note that this signal should be hardwired to 1‘b1 if the bridge is connected to an OCP master that does not implement response flow control</td><td>30%</td></tr>
<tr><td>OC_MRespAccept_En</td><td>SI</td><td>OCP master supports response flow control. (1 = supports, 0 = no support).This signal indicates if the OCP master implements OC_MRespAccept signal.</td><td>NA</td></tr>
<tr><td colspan="4">Global Signals</td></tr>
<tr><td>OC_Clk</td><td>I</td><td>Global clock signal. All signals are sampled at the rising edge of the global clock</td><td>NA</td></tr>
<tr><td>OC_MReset_n</td><td>I</td><td>Active low indication that the core is being reset and other OCP devices should be reset as well.</td><td>30%</td></tr>
<tr><td colspan="4">Miscellaneous</td></tr>
<tr><td>dflt_port_priority</td><td>SI</td><td>Default port priority. This indicates which ports (0 = Port 0, 1 = Port 1) will have the default priority. Note that the actual priority is dependent on dflt_port_priority, OC_MRespAccept_En, P0_MRespAccept_En, and P1_MRespAccept_En.</td><td>NA</td></tr>
<tr><td>SCANENABLE</td><td>I</td><td>Scan enable signal. Used to enable conditional registers during scan operation</td><td>30%</td></tr>
<tr><td>SCANMODE</td><td>I</td><td>Scan mode signal</td><td>30%</td></tr>
<tr><td>SCANIN[x:0]</td><td>I</td><td>Scan-in chains. The number of scan chains is determined by the value <code>MBB_NUM_SCAN_CHAIN</code> in the config file</td><td>30%</td></tr>
<tr><td>SCANOUT[x:0]</td><td>I</td><td>Scan-out chains. The number of scan chains is determined by the value <code>MBB_NUM_SCAN_CHAIN</code> in the config file</td><td>30%</td></tr>
<tr><td>SI_SBlock</td><td>SO</td><td>This port indicates the burst ordering mode supported by the bridge. This signal is statically driven to 0, indicating that only sequential ordering is supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows:

. L2 OCP Master: Connect to port <code>SI_L2_SBlock</code> of L2
. Core OCP Master: Connect to port <code>SI_SBlock</code> of core
. CM OCP Master: Connect to port <code>SI_SBlock</code> of CM</td><td>NA</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>SI_SimpleBE</td><td>SO</td><td>This port indicates byte enables support by the bridge. This signal is statically driven to 1, indicating that only simple byte enables are supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows:

. Core OCP Master: Connect to port <code>SI_SimpleBE</code> of core
. CM OCP Master: Connect to port <code>SI_SimpleBE</code> of CM</td><td>NA</td></tr>
<tr><td>SI_SyncTxEn</td><td>SO</td><td>This port indicates support for externalized SYNC command. This signal is statically driven to 1, indicating that the bridge supports external SYNC command. Based on what is connected at the bridge OCP master port, the port should be connected as follows:

. L2 OCP Master: Connect to port <code>L2_SyncTxEn</code> of L2
. Core OCP Master: Connect to port <code>SI_SimpleBE</code> of core
. CM OCP Master: Connect to port <code>SI_CM_SimpleBE</code> of CM</td><td>NA</td></tr>
<tr><td>SI_256b_Dp_En</td><td>SO</td><td>This port indicates support for 256 bits data mode. This signal is statically driven to 0, indicating that the bridge supports only 64 bits. This signal is used only when the CM is the OCP master. In that case, the port should be connected to port <code>SI_CM_256b_Dp_En</code> of the CM.</td><td>NA</td></tr>
<tr><td colspan="4">Port 0 Signals</td></tr>
<tr><td>P0_MCmd[2:0]</td><td>O</td><td>OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on P0_MReqInfo and P0_MAddrSpace. The encodings used by the OCP master are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Used for data write and L2 CACHE write or invalidate</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Used for fetch or data read or L2 CACHE reads or SYNC.</td></tr>
<tr><td>3-7</td><td>Unused</td><td>-</td><td>Not used</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P0_MReqInfo[6:0]</td><td>O</td><td>OCP command bus extension.
For transactions other than SYNC and CACHE, the P0_MReqInfo[2:0] field encodes the cacheability attributes for a transaction.
P0_MReqInfo[3] indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.
The encoding of the P0_MReqInfo field for all transactions other than CACHE is summarized in the following table:
Encoding for all Transactions Other Than CACHE<table><tr><th>Encoding</th><th>Command Information</th></tr><tr><td>0</td><td>Cacheable, noncoherent, WT, NWA</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>Uncached</td></tr><tr><td>3</td><td>Cacheable, noncoherent, WB, WA</td></tr><tr><td>4-6</td><td>Reserved</td></tr><tr><td>7</td><td>Uncached accelerated</td></tr><tr><td>8-9</td><td>Reserved</td></tr><tr><td>10</td><td>SYNC with uncached CCA</td></tr><tr><td>11-15</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>P0_MAddrSpace[1:0]</td><td>O</td><td>L2/L3 Address Space indicator. When the OCP master is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.

The encoding of this field is summarized in the following table:<table><tr><th>Encoding</th><th>Address Space</th></tr><tr><td>0</td><td>Normal address space</td></tr><tr><td>1</td><td>L2 address space</td></tr><tr><td>2</td><td>L3 address space</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>P0_MAddr['MBB_ADDR_WIDTH-1:0]</td><td>O</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (P0_MByteEn) or write (P0_MDataByteEn) byte enable fields.</td><td>60%</td></tr>
<tr><td>P0_MBurstSeq[2:0]</td><td>O</td><td>Indicates type of burst sequence. The OCP master can only generate two possible values, determined by the SI_SBlock static input, as shown in the following table:<table><tr><th>Encoding</th><th>Burst Sequence</th></tr><tr><td>2</td><td>Sequential: Critical dword first, with linear wrapping for subsequent beats</td></tr><tr><td>4</td><td>Sub-block Critical dword first, with increment/decrement for subsequent beats</td></tr><tr><td>0-1,3,5-7</td><td>Unused</td></tr></table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P0_MTagID_['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Transaction tag identifier.
Note: All writes have tag ID of 4‘h7</td><td>60%</td></tr>
<tr><td>P0_MBurstPrecise</td><td>O</td><td>Indicates whether the burst length is precise. Burst lengths are always fixed at 4 beats (or 8 beats based on configuration), so this pin is statically set to 0x1.</td><td>60%</td></tr>
<tr><td>P0_MBurstSingleReq</td><td>O</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>60%</td></tr>
<tr><td>P0_MBurstLength[2:0]</td><td>O</td><td>Number of 64b data transfers.
<table><tr><th>Encoding</th><th>Number of Transfers</th></tr><tr><td>1</td><td>1, single transfer</td></tr><tr><td>4</td><td>4-beat burst</td></tr><tr><td>8</td><td>8-beat burst</td></tr><tr><td>others</td><td>Unused</td></tr></table></td><td>60%</td></tr>
<tr><td>P0_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the P0_MByteEn field to the returned read data bytes is shown in the following table:
<table><tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr><tr><td>[0]</td><td>[7:0]</td></tr><tr><td>[1]</td><td>[15:8]</td></tr><tr><td>[2]</td><td>[23:16]</td></tr><tr><td>[3]</td><td>[31:24]</td></tr><tr><td>[4]</td><td>[39:32]</td></tr><tr><td>[5]</td><td>[47:40]</td></tr><tr><td>[6]</td><td>[55:48]</td></tr><tr><td>[7]</td><td>[63:56]</td></tr><tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr></table></td><td>60%</td></tr>
<tr><td>P0_MData['MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Write data bus from the OCP master</td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P0_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the P0_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use P0_MByteEn for transferring byte enables.
<table>
<tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>60%</td></tr>
<tr><td>P0_MDataValid</td><td>O</td><td>Valid write data on P0_MData bus.</td><td>60%</td></tr>
<tr><td>P0_MDataTagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Write data tag identifier (for out of order returns). All writes by the OCP master should have a TagID value of 0x7.</td><td>60%</td></tr>
<tr><td>P0_MDataLast</td><td>O</td><td>Last data in a write burst - only valid when P0_MDataValid is asserted.</td><td>60%</td></tr>
<tr><td>P0_SDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Returned read data to core.</td><td>60%</td></tr>
<tr><td>P0_STagID[3:0]</td><td>I</td><td>Return transaction tag ID. See P0_MTagID for encoding.</td><td>60%</td></tr>
<tr><td>P0_SResp[1:0]</td><td>I</td><td>Valid response from system controller. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td>Should not be used by OCP master</td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals bus error exception</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P0_SRespInfo[1:0]</td><td>I</td><td>P0_SRespInfo[0] indicates the type of error reported on P0_SResp. Valid only when P0_SResp shows a response error (value of 0x3).
<table>
<tr><th>OC_SRespInfo[0]</th><th>Description</th></tr>
<tr><td>0</td><td>Bus Error</td></tr>
<tr><td>1</td><td>Cache Error</td></tr>
</table>
P0_SRespInfo[1] indicates the cache line state of return data. L1 D-cache will install the line as the state that is specified on this field.
<table>
<tr><th>OC_SRespInfo[1]</th><th>Description</th></tr>
<tr><td>0</td><td>Clean line return</td></tr>
<tr><td>1</td><td>Modified/Dirty line return.</td></tr>
</table>
Note: This signal is statically driven to 2‘b00</td><td>60%</td></tr>
<tr><td>P0_SRespLast</td><td>I</td><td>Marks last data in read burst.</td><td>60%</td></tr>
<tr><td>P0_SCmdAccept</td><td>I</td><td>OCP-SPL notifies the OCP master that the command is accepted.</td><td>50%</td></tr>
<tr><td>P0_SDataAccept</td><td>I</td><td>OCP-SPL notifies the OCP master that the write data is accepted.</td><td>50%</td></tr>
<tr><td>P0_MRespAccept</td><td>O</td><td>OCP master accepts the read response. Note that this signal should be hardwired to 1‘b1 if the bridge is connected to an OCP master that does not implement response flow control</td><td>30%</td></tr>
<tr><td>P0_MRespAccept_En</td><td>SI</td><td>Port 0 slave supports response flow control. (1 = supports, 0 = no support).This signal indicates if the OCP slave implements P0_MRespAccept signal.</td><td>NA</td></tr>
<tr><td colspan="4">Port 1 Signals</td></tr>
<tr><td>P1_MCmd[2:0]</td><td>O</td><td>OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on P1_MReqInfo and P1_MAddrSpace. The encodings used by the OCP master are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Used for data write and L2 CACHE write or invalidate</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Used for fetch or data read or L2 CACHE reads or SYNC.</td></tr>
<tr><td>3-7</td><td>Unused</td><td>-</td><td>Not used</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P1_MReqInfo[6:0]</td><td>O</td><td>OCP command bus extension.
For transactions other than SYNC and CACHE, the P1_MReqInfo[2:0] field encodes the cacheability attributes for a transaction.
P1_MReqInfo[3] indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.
The encoding of the P1_MReqInfo field for all transactions other than CACHE is summarized in the following table:
Encoding for all Transactions Other Than CACHE<table><tr><th>Encoding</th><th>Command Information</th></tr><tr><td>0</td><td>Cacheable, noncoherent, WT, NWA</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>Uncached</td></tr><tr><td>3</td><td>Cacheable, noncoherent, WB, WA</td></tr><tr><td>4-6</td><td>Reserved</td></tr><tr><td>7</td><td>Uncached accelerated</td></tr><tr><td>8-9</td><td>Reserved</td></tr><tr><td>10</td><td>SYNC with uncached CCA</td></tr><tr><td>11-15</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>P1_MAddrSpace[1:0]</td><td>O</td><td>L2/L3 Address Space indicator. When the OCP master is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.

The encoding of this field is summarized in the following table:<table><tr><th>Encoding</th><th>Address Space</th></tr><tr><td>0</td><td>Normal address space</td></tr><tr><td>1</td><td>L2 address space</td></tr><tr><td>2</td><td>L3 address space</td></tr><tr><td>3</td><td>Reserved</td></tr></table></td><td>60%</td></tr>
<tr><td>P1_MAddr['MBB_ADDR_WIDTH-1:0]</td><td>O</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (P1_MByteEn) or write (P1_MDataByteEn) byte enable fields.</td><td>60%</td></tr>
<tr><td>P1_MBurstSeq[2:0]</td><td>O</td><td>Indicates type of burst sequence. The OCP master can only generate two possible values, determined by the SI_SBlock static input, as shown in the following table:<table><tr><th>Encoding</th><th>Burst Sequence</th></tr><tr><td>2</td><td>Sequential: Critical dword first, with linear wrapping for subsequent beats</td></tr><tr><td>4</td><td>Sub-block Critical dword first, with increment/decrement for subsequent beats</td></tr><tr><td>0-1,3,5-7</td><td>Unused</td></tr></table></td><td>60%</td></tr>
<tr><td>P1_MTagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Transaction tag identifier.
Note: All writes have tag ID of 4‘h7</td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P1_MBurstPrecise</td><td>O</td><td>Indicates whether the burst length is precise. Burst lengths are always fixed at 4 beats (or 8 beats based on configuration), so this pin is statically set to 0x1.</td><td>60%</td></tr>
<tr><td>P1_MBurstSingleReq</td><td>O</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>60%</td></tr>
<tr><td>P1_MBurstLength[2:0]</td><td>O</td><td>Number of 64b data transfers.
<table><tr><th>Encoding</th><th>Number of Transfers</th></tr><tr><td>1</td><td>1, single transfer</td></tr><tr><td>4</td><td>4-beat burst</td></tr><tr><td>8</td><td>8-beat burst</td></tr><tr><td>others</td><td>Unused</td></tr></table></td><td>60%</td></tr>
<tr><td>P1_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the P1_MByteEn field to the returned read data bytes is shown in the following table:
<table><tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr><tr><td>[0]</td><td>[7:0]</td></tr><tr><td>[1]</td><td>[15:8]</td></tr><tr><td>[2]</td><td>[23:16]</td></tr><tr><td>[3]</td><td>[31:24]</td></tr><tr><td>[4]</td><td>[39:32]</td></tr><tr><td>[5]</td><td>[47:40]</td></tr><tr><td>[6]</td><td>[55:48]</td></tr><tr><td>[7]</td><td>[63:56]</td></tr><tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr></table></td><td>60%</td></tr>
<tr><td>P1_MData['MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Write data bus from the OCP master</td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P1_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the P1_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use P1_MByteEn for transferring byte enables.
<table>
<tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>60%</td></tr>
<tr><td>P1_MDataValid</td><td>O</td><td>Valid write data on P1_MData bus.</td><td>60%</td></tr>
<tr><td>P1_MDataTagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Write data tag identifier (for out of order returns). All writes by the OCP master should have a TagID value of 0x7.</td><td>60%</td></tr>
<tr><td>P1_MDataLast</td><td>O</td><td>Last data in a write burst - only valid when P1_MDataValid is asserted.</td><td>60%</td></tr>
<tr><td>P1_SDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Returned read data to core.</td><td>60%</td></tr>
<tr><td>P1_STagID[3:0]</td><td>I</td><td>Return transaction tag ID. See P1_MTagID for encoding.</td><td>60%</td></tr>
<tr><td>P1_SResp[1:0]</td><td>I</td><td>Valid response from system controller. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td>Should not be used by OCP master</td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals bus error exception</td></tr>
</table></td><td>60%</td></tr>
</table>

Table 8.3 OCP-SPL Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>P1_SRespInfo[1:0]</td><td>I</td><td>P1_SRespInfo[0] indicates the type of error reported on P1_SResp. Valid only when P1_SResp shows a response error (value of 0x3).
<table>
<tr><th>OC_SRespInfo[0]</th><th>Description</th></tr>
<tr><td>0</td><td>Bus Error</td></tr>
<tr><td>1</td><td>Cache Error</td></tr>
</table>
P1_SRespInfo[1] indicates the cache line state of return data. L1 D-cache will install the line as the state that is specified on this field.
<table>
<tr><th>OC_SRespInfo[1]</th><th>Description</th></tr>
<tr><td>0</td><td>Clean line return</td></tr>
<tr><td>1</td><td>Modified/Dirty line return.</td></tr>
</table>
Note: This signal is statically driven to 2‘b00</td><td>60%</td></tr>
<tr><td>P1_SRespLast</td><td>I</td><td>Marks last data in read burst.</td><td>60%</td></tr>
<tr><td>P1_SCmdAccept</td><td>I</td><td>OCP-SPL notifies the OCP master that the command is accepted.</td><td>50%</td></tr>
<tr><td>P1_SDataAccept</td><td>I</td><td>OCP-SPL notifies the OCP master that the write data is accepted.</td><td>50%</td></tr>
<tr><td>P1_MRespAccept</td><td>O</td><td>OCP master accepts the read response. Note that this signal should be hardwired to 1‘b1 if the bridge is connected to an OCP master that does not implement response flow control</td><td>30%</td></tr>
<tr><td>P1_MRespAccept_En</td><td>SI</td><td>Port 0 slave supports response flow control. (1 = supports, 0 = no support).This signal indicates if the OCP slave implements P1_MRespAccept signal.</td><td>NA</td></tr>
</table>

8.4 OCP-AXI2 Detailed Signal Descriptions

Table 8.3 lists the pinout of the OCP-AXI2. The OCP interfaces are not fully registered. However, all signals are synchronous to the rising edge of the primary OCP clock, *OC_Clk.* Outputs on the interface may have an amount of logic after the preceding flop(s), and inputs may go through some combinational logic before being registered by the bridge. Other signals pass though some combinational logic in the bridge before going out to output ports.

The expression of timing constraints for the OCP-AXI2 depends on many factors, such as maximum target frequency, process technology, standard cell library characteristics, etc., so it is difficult to provide a generic set of timing guidelines that will apply in all situations. The “Timing” column in Table 8.3 shows the timing of the OCP-AXI2 interface signals, expressed as a percentage of the minimum target period. All the input and output directions are with respect to the OCP-AXI2. For an output, the timing number means percentage of the cycle after the driving clock edge when the data is available. For an input, the number means percentage of the cycle from the preceding clock edge when data is required.

Table 8.4 OCP-AXI2 Signals

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td colspan="4">OCP Signals</td></tr>
<tr><td>OC_MCmd[2:0]</td><td>I</td><td>OCP command bus, indicates the type of transaction requested. Only some encodings are used and they are set in concert with the values on OC_MReqInfo and OC_MAddrSpace. The encodings used by the OCP master are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Used for data write and L2 CACHE write or invalidate</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Used for fetch or data read or L2 CACHE reads or SYNC.</td></tr>
<tr><td>3-7</td><td>Unused</td><td>-</td><td>Not used</td></tr>
</table></td><td>75%</td></tr>
<tr><td>OC_MReqInfo[6:0]</td><td>I</td><td>OCP command bus extension.
For transactions other than SYNC and CACHE, the OC_MReqInfo[2:0] field encodes the cacheability attributes for a transaction.
OC_MReqInfo[3] indicates that the transaction is due to a SYNC instruction; when this bit is high, the lower bits [2:0] indicate an uncached CCA type.
The encoding of the OC_MReqInfo field for all transactions other than CACHE is summarized in the following table:
Encoding for all Transactions Other Than CACHE
<table>
<tr><th>Encoding</th><th>Command Information</th></tr>
<tr><td>0</td><td>Cacheable, noncoherent, WT, NWA</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>2</td><td>Uncached</td></tr>
<tr><td>3</td><td>Cacheable, noncoherent, WB, WA</td></tr>
<tr><td>4-6</td><td>Reserved</td></tr>
<tr><td>7</td><td>Uncached accelerated</td></tr>
<tr><td>8-9</td><td>Reserved</td></tr>
<tr><td>10</td><td>SYNC with uncached CCA</td></tr>
<tr><td>11-15</td><td>Reserved</td></tr>
</table>
The AXI bridge does not support L2/L3 commands, so only the above encodings are recognized. OC_MReqInfo[6:4] field is used for L2 cache exclusivity control, and as such are ignored by the AXI bridge.</td><td>50%</td></tr>
</table>

Table 8.4 OCP-AXI2 Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MAddrSpace[1:0]</td><td>I</td><td>L2/L3 Address Space indicator. When the OCP master is issuing an L2 or an L3 CACHE operation, the corresponding bit (Bit [0] for L2, and Bit [1] for L3) is asserted. It indicates to the system that this OCP command is targeted to the address space of the L2 or L3 Cache.
Note: The AXI bridge does not support L2/L3 commands.
The encoding of this field is summarized in the following table:
<table>
<tr><th>Encoding</th><th>Address Space</th></tr>
<tr><td>0</td><td>Normal address space</td></tr>
<tr><td>1</td><td>L2 address space</td></tr>
<tr><td>2</td><td>L3 address space</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table></td><td>50%</td></tr>
<tr><td>OC_MAddr['MBB_ADDR_WIDTH-1:0]</td><td>I</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (OC_MByteEn) or write (OC_MDataByteEn) byte enable fields.</td><td>70%</td></tr>
<tr><td>OC_MBurstSeq[2:0]</td><td>I</td><td>Indicates type of burst sequence. The OCP master can only generate two possible values, determined by the SI_SBlock static input, as shown in the following table:
<table>
<tr><th>Encoding</th><th>Burst Sequence</th></tr>
<tr><td>2</td><td>Sequential: Critical dword first, with linear wrapping for subsequent beats</td></tr>
<tr><td>4</td><td>Sub-block Critical dword first, with increment/decrement for subsequent beats</td></tr>
<tr><td>0-1,3,5-7</td><td>Unused</td></tr>
</table>
Note: The AXI bridge does not support sub-block ordering. SI_SBlock must be hardwired to 0.</td><td>Unused</td></tr>
<tr><td>OC_MTagID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Transaction tag identifier.
Note: All writes have tag ID of 4'h7</td><td>70%</td></tr>
<tr><td>OC_MBurstPrecise</td><td>SI</td><td>Indicates whether the burst length is precise. Burst lengths are always fixed at 4 beats (or 8 beats based on configuration), so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstSingleReq</td><td>SI</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>Unused</td></tr>
<tr><td>OC_MBurstLength[2:0]</td><td>I</td><td>Number of 64b data transfers.
<table>
<tr><th>Encoding</th><th>Number of Transfers</th></tr>
<tr><td>1</td><td>1, single transfer</td></tr>
<tr><td>4</td><td>4-beat burst</td></tr>
<tr><td>8</td><td>8-beat burst</td></tr>
<tr><td>others</td><td>Unused</td></tr>
</table></td><td>45%</td></tr>
</table>

Table 8.4 OCP-AXI2 Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the OC_MByteEn field to the returned read data bytes is shown in the following table:
<table><tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr><tr><td>[0]</td><td>[7:0]</td></tr><tr><td>[1]</td><td>[15:8]</td></tr><tr><td>[2]</td><td>[23:16]</td></tr><tr><td>[3]</td><td>[31:24]</td></tr><tr><td>[4]</td><td>[39:32]</td></tr><tr><td>[5]</td><td>[47:40]</td></tr><tr><td>[6]</td><td>[55:48]</td></tr><tr><td>[7]</td><td>[63:56]</td></tr><tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr></table></td><td>65%</td></tr>
<tr><td>OC_MCmdSideBand[MBB_SIDEBAND_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MCmd. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification, and is used to pass user-defined signals to the AXI domain (see ARSIDEBAND and AWIDEBAND).</td><td>35%</td></tr>
<tr><td>OC_MConnID[`MBB_O2A_CONNID_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MCmd. The size of the signal is configurableby setting the value of ‘MBB_O2A_CONNID_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification and is used to pass user-defined signals to the AXI domain (see ARSIDEBAND and AWIDEBAND).</td><td>35%</td></tr>
<tr><td>OC_MDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Write data bus from the OCP master</td><td>45%</td></tr>
<tr><td>OC_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>I</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the OC_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use OC_MByteEn for transferring byte enables.
<table><tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr><tr><td>[0]</td><td>[7:0]</td></tr><tr><td>[1]</td><td>[15:8]</td></tr><tr><td>[2]</td><td>[23:16]</td></tr><tr><td>[3]</td><td>[31:24]</td></tr><tr><td>[4]</td><td>[39:32]</td></tr><tr><td>[5]</td><td>[47:40]</td></tr><tr><td>[6]</td><td>[55:48]</td></tr><tr><td>[7]</td><td>[63:56]</td></tr><tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr></table></td><td>45%</td></tr>
</table>

Table 8.4 OCP-AXI2 Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MDataValid</td><td>I</td><td>Valid write data on OC_MData bus.</td><td>25%</td></tr>
<tr><td>OC_MDataTagID[‘MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Write data tag identifier (for out of order returns). All writes by the OCP master should have a TagID value of 0x7.</td><td>10%</td></tr>
<tr><td>OC_MDataLast</td><td>I</td><td>Last data in a write burst - only valid when OC_MDataValid is asserted.</td><td>10%</td></tr>
<tr><td>OC_MDataSideBand[MBB_SIDEBAND_WIDTH:0]</td><td>I</td><td>Sideband signal associated with OC_MData. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification, and is used to pass user defined signals to the AXI domain (See WSIDEBAND).</td><td>10%</td></tr>
<tr><td>OC_MReset_n</td><td>I</td><td>Active low indication that the core is being reset and other OCP devices should be reset as well.</td><td>30%</td></tr>
<tr><td>OC_SDATA[‘MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Returned read data to core.</td><td>55%</td></tr>
<tr><td>OC_STagID[‘MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Return transaction tag ID. See OC_MTagID for encoding.</td><td>55%</td></tr>
<tr><td>OC_SResp[1:0]</td><td>O</td><td>Valid response from system controller. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td>Should not be used by OCP master</td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals bus error exception</td></tr>
</table></td><td>60%</td></tr>
<tr><td>OC_SRespInfo[1:0]</td><td>SO</td><td>OC_SRespInfo[0] indicates the type of error reported on OC_SResp. Valid only when OC_SResp shows a response error (value of 0x3).
<table>
<tr><th>OC_SRespInfo[0]</th><th>Description</th></tr>
<tr><td>0</td><td>Bus Error</td></tr>
<tr><td>1</td><td>Cache Error</td></tr>
</table>
OC_SRespInfo[1] indicates the cache line state of return data. L1 D-cache will install the line as the state that is specified on this field.
<table>
<tr><th>OC_SRespInfo[1]</th><th>Description</th></tr>
<tr><td>0</td><td>Clean line return</td></tr>
<tr><td>1</td><td>Modified/Dirty line return.</td></tr>
</table>
Note: This signal is statically driven to 2‘b00</td><td>NA</td></tr>
<tr><td>OC_SRespLast</td><td>O</td><td>Marks last data in read burst.</td><td>55%</td></tr>
<tr><td>OC_SCmdAccept</td><td>O</td><td>AXI bridge notifies the OCP master that the command is accepted.</td><td>50%</td></tr>
<tr><td>OC_SDataAccept</td><td>O</td><td>AXI bridge notifies the OCP master that the write data is accepted.</td><td>55%</td></tr>
<tr><td>OC_MRespAccept</td><td>I</td><td>OCP master accepts the read response. Note that this signal should be hardwired to 1‘b1 if the bridge is connected to an OCP master that does not implement response flow control</td><td>55%</td></tr>
<tr><td>OC_MRespAccept_En</td><td>SI</td><td>OCP master supports response flow control (1 = supports, 0 = no support).This signal indicates if the OCP master implements OC_MRespAccept signal.</td><td>NA</td></tr>
</table>

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
Miscellaneous			
dflt_port_priority	SI	Default port priority. This indicates which ports (0 = Port 0, 1 = Port 1) will have the default priority. Note that the actual priority is dependent on *dflt_port_priority*, *OC_MRespAccept_En*, *P0_MRespAccept_En*, and *P1_MRespAccept_En*.	NA
P0_MRespAccept_En	SI	Port 0 supports response flow control. (1 = supports, 0 = no support).This signal **MUST** be connected to 1‘b1.	NA
P1_MRespAccept_En	SI	Port 1supports response flow control. (1 = supports, 0 = no support).This signal **MUST** be connected to 1‘b1.	NA
SCANENABLE	I	Scan enable signal. Used to enable conditional registers during scan operation	30%
SCANMODE	I	Scan mode signal	30%
SCANIN[x:0]	I	Scan-in chains. The number of scan chains is determined by the value `MBB_NUM_SCAN_CHAIN` in the config file	30%
SCANOUT[x:0]	I	Scan-out chains. The number of scan chains is determined by the value `MBB_NUM_SCAN_CHAIN` in the config file	30%
P0_AXI_WERR_INT	O	Write error interrupt signal for port 0. This signal is asserted when the AXI bridge decodes an error write response.	15%
P0_AXI_WERR_ADDR ['MBB_ADDR_WIDTH-1:0]	O	Write error address signal for port 0. This signal has the doubleword address of the write that caused the error response.	15%
P0_AXI_WERR_TYPE	O	Write error type signal for port 0. This signal encodes the error type (0 = slave error, 1 = address decode error)	15%
P0_AXI_WERR_ACK	I	Write error interrupt acknowledge for port 0. This signal is used to de-assert the *AXI_WERR_INT* signal.	25%
P0_AXI_CMD_PROT[1:0]	I	Command protection bits for port 0. These bits map directly to the privileged/secure bits in *P0_ARPROT[1:0]* and *P0_AWPROT[1:0]*. They are included since there is no equivalent signals on the OCP side. The user may hardwire the signal to some default value, or include some external logic to generate them based on the OCP command info. Note that the signal associates with the current command on the OCP bus.	20%
P1_AXI_WERR_INT	O	Write error interrupt signal for port 0. This signal is asserted when the AXI bridge decodes an error write response.	15%
P1_AXI_WERR_ADDR ['MBB_ADDR_WIDTH-1:0]	O	Write error address signal for port 0. This signal has the doubleword address of the write that caused the error response.	15%
P1_AXI_WERR_TYPE	O	Write error type signal for port 0. This signal encodes the error type (0 = slave error, 1 = address decode error)	15%
P1_AXI_WERR_ACK	I	Write error interrupt acknowledge for port 0. This signal is used to de-assert the *AXI_WERR_INT* signal.	25%
P1_AXI_CMD_PROT[1:0]	I	Command protection bits for port 0. These bits map directly to the privileged/secure bits in *P1_ARPROT[1:0]* and *P1_AWPROT[1:0]*. They are included since there is no equivalent signals on the OCP side. The user may hardwire the signal to some default value, or include some external logic to generate them based on the OCP command info. Note that the signal associates with the current command on the OCP bus.	20%

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
SI_SBlock	SO	This port indicates the burst ordering mode supported by the bridge. This signal is statically driven to 0, indicating that only sequential ordering is supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . L2 OCP Master: Connect to port *SI_L2_SBlock* of L2 . Core OCP Master: Connect to port *SI_SBlock* of core . CM OCP Master: Connect to port *SI_SBlock* of CM	NA
SI_SimpleBE	SO	This port indicates byte enables support by the bridge. This signal is statically driven to 1, indicating that only simple byte enables are supported. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . Core OCP Master: Connect to port *SI_SimpleBE* of core . CM OCP Master: Connect to port *SI_SimpleBE* of CM	NA
SI_SyncTxEn	SO	This port indicates support for externalized SYNC command. This signal is statically driven to 1, indicating that the bridge supports external SYNC command. Based on what is connected at the bridge OCP master port, the port should be connected as follows: . L2 OCP Master: Connect to port *L2_SyncTxEn* of L2 . Core OCP Master: Connect to port *SI_SimpleBE* of core . CM OCP Master: Connect to port *SI_CM_SimpleBE* of CM	NA
SI_256b_Dp_En	SO	This port indicates support for 256 bits data mode. This signal is statically driven to 0, indicating that the bridge supports only 64 bits. This signal is used only when the CM is the OCP master. In that case, the port should be connected to port *SI_CM_256b_Dp_En* of the CM.	NA
Global Signals			
ACLK	I	Global clock signal. All signals are sampled at the rising edge of the global clock	NA
ARESETn	I	Global reset signal. This signal is active LOW	30%
AXI Signals (Port 0)			
Write Address Channel			
P0_AWID['MBB_TAGID_WIDTH-1:0]	O	Write address ID. This signal is the identification tag for the write address group of signals	15%
P0_AWADDR['MBB_ADDR_WIDTH-1:0]	O	Write address. The write address bus gives the address of the first transfer in a write burst transaction	15%
P0_AWLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst	15%
P0_AWSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst	15%
P0_AWBURST[1:0]	O	Burst type	15%
P0_AWLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer	15%
P0_AWCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction	15%

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
P0_AWPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access	15%
P0_AWVALID	O	Write address valid. This signal indicates that valid write address and control information is available.	15%
P0_AWSIDEBAND['MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the write command. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
P0_AWREADY	I	Write address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Write Data Channel			
P0_WID['MBB_TAGID_WIDTH-1:0]	O	Write ID tag. this signal is the ID tag of the write data transfer	15%
*P0_WDATA['MBB_BUS_DEFTYPE*64-1:0]*	O	Write data	15%
*P0_WSTRB['MBB_BUS_DEFTYPE*8-1:0]*	O	Write strobes. This signal indicates which byte lanes to update in memory	15%
P0_WLAST	O	Write last. This signal indicates the last transfer in a write burst	15%
P0_WVALID	O	Write valid. This signal indicates that valid write data and strobes are available	15%
P0_WSIDEBAND['MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the write data. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MDataSideBand*).	15%
P0_WREADY	I	Write ready. This signal indicates that the slave can accept the write data	30%
Write Response Channel			
P0_BID['MBB_TAGID_WIDTH-1:0]	I	Response ID. The identification tag of the write response	Unused
P0_BRESP[1:0]	I	Write response. This signal indicates the status of the write transaction	25%
P0_BVALID	I	Write response valid. This signal indicates that a valid write response is available	25%
P0_BREADY	SO	Response ready. This signal indicates that the AXI bridge can accept the response information. Note that the AXI bridge drives this signal statically to a value of 1	NA
Read Address Channel			
P0_ARID['MBB_TAGID_WIDTH-1:0]	O	Read address ID. This signal is the identification tag for the Read address group of signals	15%
P0_ARADDR['MBB_ADDR_WIDTH-1:0]	O	Read address. The Read address bus gives the address of the first transfer in a Read burst transaction	15%
P0_ARLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst	15%
P0_ARSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst	15%
P0_ARBURST[1:0]	O	Burst type	15%
P0_ARLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer	15%

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
P0_ARCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction	15%
P0_ARPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access	15%
P0_ARVALID	O	Read address valid. This signal indicates that valid Read address and control information is available.	15%
P0_ARSIDEBAND[`MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the read command. The size of the signal is configurable by setting the value of ‘`MBB_SIDEBAND_WIDTH` define in the configuration file (`mbb_config.vh`). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
P0_ARREADY	I	Read address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Read Data Channel			
P0_RID['MBB_TAGID_WIDTH-1:0]	I	Read ID tag. This signal is the ID tag of the read data group of signals	55%
*P0_RDATA['MBB_BUS_DEFTYPE*64-1:0]*	I	Read data	55%
P0_RRESP[1:0]	I	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR	60%
P0_RLAST	I	Read last. This signal indicates the last transfer in a read burst	55%
P0_RVALID	I	Read valid. This signal indicates that the required read data is available and the read transfer can complete	60%
P0_RREADY	O	Read ready. This signal indicates that the master can accept the read data and response information.	55%
AXI Signals (Port 1)			
Write Address Channel			
P1_AWID['MBB_TAGID_WIDTH-1:0]	O	Write address ID. This signal is the identification tag for the write address group of signals	15%
P1_AWADDR['MBB_ADDR_WIDTH-1:0]	O	Write address. The write address bus gives the address of the first transfer in a write burst transaction	15%
P1_AWLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst	15%
P1_AWSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst	15%
P1_AWBURST[1:0]	O	Burst type	15%
P1_AWLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer	15%
P1_AWCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction	15%
P1_AWPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access	15%
P1_AWVALID	O	Write address valid. This signal indicates that valid write address and control information is available.	15%

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
P1_AWSIDEBAND[`MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the write command. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
P1_AWREADY	I	Write address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Write Data Channel			
P1_WID[‘MBB_TAGID_WIDTH-1:0]	O	Write ID tag. this signal is the ID tag of the write data transfer	15%
*P1_WDATA[‘MBB_BUS_DEFTYPE*64-1:0]*	O	Write data	15%
*P1_WSTRB[‘MBB_BUS_DEFTYPE*8-1:0]*	O	Write strobes. This signal indicates which byte lanes to update in memory	15%
P1_WLAST	O	Write last. This signal indicates the last transfer in a write burst	15%
P1_WVALID	O	Write valid. This signal indicates that valid write data and strobes are available	15%
P1_WSIDEBAND[`MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the write data. The size of the signal is configurable by setting the value of ‘MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MDataSideBand*).	15%
P1_WREADY	I	Write ready. This signal indicates that the slave can accept the write data	30%
Write Response Channel			
P1_BID[‘MBB_TAGID_WIDTH-1:0]	I	Response ID. The identification tag of the write response	Unused
P1_BRESP[1:0]	I	Write response. This signal indicates the status of the write transaction	25%
P1_BVALID	I	Write response valid. This signal indicates that a valid write response is available	25%
P1_BREADY	SO	Response ready. This signal indicates that the AXI bridge can accept the response information. Note that the AXI bridge drives this signal statically to a value of 1	NA
Read Address Channel			
P1_ARID[‘MBB_TAGID_WIDTH-1:0]	O	Read address ID. This signal is the identification tag for the Read address group of signals	15%
P1_ARADDR[‘MBB_ADDR_WIDTH-1:0]	O	Read address. The Read address bus gives the address of the first transfer in a Read burst transaction	15%
P1_ARLEN[3:0]	O	Burst Length. The burst length gives the exact number of transfers in a burst	15%
P1_ARSIZE[2:0]	O	Burst size. This signal indicates the size of each transfer in the burst	15%
P1_ARBURST[1:0]	O	Burst type	15%
P1_ARLOCK[1:0]	O	Lock type. This signal provides additional information about the atomic characteristics of the transfer	15%
P1_ARCACHE[3:0]	O	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction	15%
P1_ARPROT[2:0]	O	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access	15%

Table 8.4 OCP-AXI2 Signals (Continued)

Signal Name	Dir	Description	Timing
P1_ARVALID	O	Read address valid. This signal indicates that valid Read address and control information is available.	15%
P1_ARSIDEBAND[`MBB_SIDEBAND_WIDTH:0]	O	Sideband signal associated with the read command. The size of the signal is configurable by setting the value of ‘`MBB_SIDEBAND_WIDTH` define in the configuration file (`mbb_config.vh`). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	15%
P1_ARREADY	I	Read address ready. This signal indicates that the slave is ready to accept and address and associated control information.	30%
Read Data Channel			
P1_RID['MBB_TAGID_WIDTH-1:0]	I	Read ID tag. This signal is the ID tag of the read data group of signals	55%
*P1_RDATA['MBB_BUS_DEFTYPE*64-1:0]*	I	Read data	55%
P1_RRESP[1:0]	I	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR	60%
P1_RLAST	I	Read last. This signal indicates the last transfer in a read burst	55%
P1_RVALID	I	Read valid. This signal indicates that the required read data is available and the read transfer can complete	60%
P1_RREADY	O	Read ready. This signal indicates that the master can accept the read data and response information.	55%

8.5 AXI-OCP Detailed Signal Descriptions

Table 8.2 lists the pinout of the AXI to OCP bridge. The OCP and AXI interfaces are not fully registered. However, all signals are synchronous to the rising edge of the primary AXI clock, *ACLK.* Outputs on the interface may have an amount of logic after the preceding flop(s), and inputs may go through some combinational logic before being registered by the bridge. Other signals pass though some combinational logic in the bridge before going out to output ports.

The expression of timing constraints for the AXI to OCP bridge depends on many factors, such as maximum target frequency, process technology, standard cell library characteristics, etc., so it is difficult to provide a generic set of timing guidelines that will apply in all situations. The “Timing” column in Table 8.2 shows the timing of the AXI to OCP bridge interface signals, expressed as a percentage of the minimum target period. All the input and output directions are with respect to the AXI to OCP bridge. For an output, the timing number means percentage of the cycle after the driving clock edge when the data is available. For an input, the number means percentage of the cycle from the preceding clock edge when data is required.

Table 8.5 AXI-OCP Bridge Signals

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td colspan="4">OCP Signals</td></tr>
<tr><td>OC_MReset_n</td><td>O</td><td>OCP master interface reset. The AXI interface reset input (ARESETn) is propagated onto the OCP interface</td><td>50%</td></tr>
<tr><td>OC_MCmd[2:0]</td><td>O</td><td>OCP command bus, indicates the type of transaction requested. The encodings used by the AXI-OCP bridge are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>Idle</td><td>IDLE</td><td>No transaction</td></tr>
<tr><td>1</td><td>Write</td><td>WR</td><td>Write</td></tr>
<tr><td>2</td><td>Read</td><td>RD</td><td>Read</td></tr>
<tr><td>5</td><td>Non-Posted Write</td><td>WRNP</td><td>Non-Posted Write</td></tr>
<tr><td>3,4,6,7</td><td>-</td><td>-</td><td>Not used</td></tr>
</table></td><td>20%</td></tr>
<tr><td>OC_MReqInfo[5:0]</td><td>O</td><td>OCP command bus extension. The width of this signal is configurable by modifying the ‘MBB_A2O_MREQINFO_WIDTH define in the configuration file (mbb_config.vh). This signal is only used when AXI-OCP is connected to the IOCU in the 1004K CPS.
This is driven by the command sideband signals from AXI. See Section 3.4.8 “Sideband Signals” on how the usage model for this signal.</td><td>20%</td></tr>
<tr><td>OC_MAddr['MBB_ADDR_WIDTH-1:0]</td><td>O</td><td>Physical doubleword address bus. Note that the least-significant 3 address bits are statically tied to 0, and the address of the byte(s) within the doubleword are indicated by the read (OC_MByteEn) or write (OC_MDataByteEn) byte enable fields.</td><td>20%</td></tr>
<tr><td>OC_MBurstSeq[2:0]</td><td>O</td><td>Indicates type of burst sequence. The OCP interface can only generate two possible values as shown in the following table:
<table>
<tr><th>Encoding</th><th>Burst Sequence</th></tr>
<tr><td>0</td><td>Incrementing: Address is incremented by OCP word size (8 bytes) each transfer.</td></tr>
<tr><td>2</td><td>Wrapping: Critical dword first, with linear wrapping for subsequent beats</td></tr>
<tr><td>1,3-7</td><td>Unused</td></tr>
</table></td><td>20%</td></tr>
<tr><td>OC_MTagID[`MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Transaction tag identifier.</td><td>20%</td></tr>
<tr><td>OC_MBurstPrecise</td><td>SO</td><td>Burst lengths are always known at the start of the burst, so this pin is statically set to 0x1.</td><td>NA</td></tr>
<tr><td>OC_MBurstSingleReq</td><td>SO</td><td>Indicates whether there is a single request for all data transfers in a burst. There is always a single command request so this pin is statically set to 0x1.</td><td>NA</td></tr>
<tr><td>OC_MBurstLength[4:0]</td><td>O</td><td>This signal indicates the number of 64 bit data transfers. Transfer lengths from 1 through 16 are supported.</td><td>20%</td></tr>
</table>

Table 8.5 AXI-OCP Bridge Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_MByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for reads. Includes data alignment, endianness and address. The correlation of each bit in the OC_MByteEn field to the returned read data bytes is shown in the following table:
<table>
<tr><th>OC_MByteEn</th><th>Requested byte to be returned on OC_SData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MByteEn</td></tr>
</table></td><td>20%</td></tr>
<tr><td>OC_MConnID[3:0]</td><td>O</td><td>Sideband signal associated with OC_MCmd. The size of the signal is configurable by setting the value of ‘MBB_O2A_MCONNID_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the OCP specification, and is used to pass user defined signals to the AXI domain (see ARSIDEBAND and AWIDEBAND).</td><td>20%</td></tr>
<tr><td>OC_MDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>O</td><td>Write data bus from the OCP master</td><td>20%</td></tr>
<tr><td>OC_MDataByteEn['MBB_BUS_DEFTYPE*8-1:0]</td><td>O</td><td>Byte enables for writes. Includes data alignment, endianness and address. The correlation of each bit in the OC_MDataByteEn field to the write data bytes is shown in the following table. Note that the OCP master should not use OC_MByteEn for transferring byte enables for Writes.
<table>
<tr><th>OC_MDataByteEn</th><th>Valid write data byte on OC_MData bus</th></tr>
<tr><td>[0]</td><td>[7:0]</td></tr>
<tr><td>[1]</td><td>[15:8]</td></tr>
<tr><td>[2]</td><td>[23:16]</td></tr>
<tr><td>[3]</td><td>[31:24]</td></tr>
<tr><td>[4]</td><td>[39:32]</td></tr>
<tr><td>[5]</td><td>[47:40]</td></tr>
<tr><td>[6]</td><td>[55:48]</td></tr>
<tr><td>[7]</td><td>[63:56]</td></tr>
<tr><td>[n]</td><td>[n*9-1:n*8]
where n is the nth bit in OC_MDataByteEn</td></tr>
</table></td><td>20%</td></tr>
<tr><td>OC_MDataValid</td><td>O</td><td>Valid write data on OC_MData bus.</td><td>20%</td></tr>
<tr><td>OC_MDataTagID['MBB_TAGID_WIDTH-1:0]</td><td>O</td><td>Write data tag identifier. This should be the same as the TagID that was used for the WRITE command.</td><td>20%</td></tr>
<tr><td>OC_MDataLast</td><td>O</td><td>Last data in a write burst - only valid when OC_MDataValid is asserted.</td><td>20%</td></tr>
</table>

Table 8.5 AXI-OCP Bridge Signals (Continued)

<table>
<tr><th>Signal Name</th><th>Dir</th><th>Description</th><th>Timing</th></tr>
<tr><td>OC_SDATA['MBB_BUS_DEFTYPE*64-1:0]</td><td>I</td><td>Returned read data to core.</td><td>70%</td></tr>
<tr><td>OC_STagID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Return transaction tag ID.</td><td>70%</td></tr>
<tr><td>OC_SResp[1:0]</td><td>I</td><td>Valid response from OCP slave. The encoding recognized are shown in the following table:
<table>
<tr><th>Encoding</th><th>Command</th><th>Mnemonic</th><th>Description</th></tr>
<tr><td>0</td><td>No response</td><td>NULL</td><td>No response</td></tr>
<tr><td>1</td><td>Data valid/accept</td><td>DVA</td><td>Normal completion response</td></tr>
<tr><td>2</td><td>Reserved</td><td>-</td><td></td></tr>
<tr><td>3</td><td>Response error</td><td>ERR</td><td>Signals an error condition</td></tr>
</table></td><td>70%</td></tr>
<tr><td>OC_SRespLast</td><td>I</td><td>Marks last data in read burst.</td><td>70%</td></tr>
<tr><td>OC_SCmdAccept</td><td>I</td><td>This OCP flow control signal indicates that the command has been accepted by the OCP slave.</td><td>50%</td></tr>
<tr><td>OC_SDataAccept</td><td>I</td><td>This OCP flow control signal indicates that the write data has been accepted by the OCP slave.</td><td>50%</td></tr>
<tr><td>OC_MRespAccept</td><td>O</td><td>This signal indicates that the AXI-OCP accepts the response returned by the OCP slave. This is used only if the AXI-OCP bridge is connected to the IOCU of the 1004K CPS. It is used if the AXI-OCP bridge is connected to the DMA interface of the ScratchPad RAMs on MIPS32 cores, since the response is expected to be accepted in the same cycle. The AXI master should ensure that RREADY and BREADY are always 1 in this case.</td><td>65%</td></tr>
<tr><td colspan="4">AXI Signals</td></tr>
<tr><td colspan="4">Global Signals</td></tr>
<tr><td>ACLK</td><td>I</td><td>Global clock signal. All signals are sampled at the rising edge of the global clock</td><td>NA</td></tr>
<tr><td>ARESETn</td><td>I</td><td>Global reset signal. This signal is active LOW</td><td>60%</td></tr>
<tr><td colspan="4">Write Address Channel</td></tr>
<tr><td>AWID['MBB_TAGID_WIDTH-1:0]</td><td>I</td><td>Write address ID. This signal is the identification tag for the write address group of signals</td><td>30%</td></tr>
<tr><td>AWADDR['MBB_ADDR_WIDTH-1:0]</td><td>I</td><td>Write address. The write address bus gives the address of the first transfer in a write burst transaction</td><td>30%</td></tr>
<tr><td>AWLEN[3:0]</td><td>I</td><td>Burst Length. The burst length gives the exact number of transfers in a burst</td><td>30%</td></tr>
<tr><td>AWSIZE[2:0]</td><td>I</td><td>Burst size. This signal indicates the size of each transfer in the burst</td><td>30%</td></tr>
<tr><td>AWBURST[1:0]</td><td>I</td><td>Burst type</td><td>30%</td></tr>
<tr><td>AWLOCK[1:0]</td><td>I</td><td>Lock type. This signal provides additional information about the atomic characteristics of the transfer</td><td>Unused</td></tr>
<tr><td>AWCACHE[3:0]</td><td>I</td><td>Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction</td><td>Unused</td></tr>
<tr><td>AWPROT[2:0]</td><td>I</td><td>Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access</td><td>Unused</td></tr>
</table>

Table 8.5 AXI-OCP Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
AWVALID	I	Write address valid. This signal indicates that valid write address and control information is available.	50%
AWSIDEBAND['MBB_O2A_CONNIDSIDEBAND_WIDTH:0]	O	Sideband signal associated with the write command. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH and 'MBB_O2A_CONNID_WIDTH defines in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	30%
AWREADY	O	Write address ready. This signal indicates that the slave is ready to accept and address and associated control information.	50%
Write Data Channel			
WID['MBB_TAGID_WIDTH-1:0]	I	Write ID tag. This signal is the ID tag of the write data transfer	30%
*WDATA['MBB_BUS_DEFTYPE*64-1:0]*	I	Write data	30%
*WSTRB['MBB_BUS_DEFTYPE*8-1:0]*	I	Write strobes. This signal indicates which byte lanes to update in memory	30%
WLAST	I	Write last. This signal indicates the last transfer in a write burst	30%
WVALID	I	Write valid. This signal indicates that valid write data and strobes are available	50%
WSIDEBAND['MBB_SIDEBAND_WIDTH:0]	I	Sideband signal associated with the write data. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH define in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (See *OC_MDataSideBand*)	30%
WREADY	O	Write ready. This signal indicates that the slave can accept the write data	50%
Write Response Channel			
BID['MBB_TAGID_WIDTH-1:0]	O	Response ID. The identification tag of the write response	55%
BRESP[1:0]	O	Write response. This signal indicates the status of the write transaction	55%
BVALID	O	Write response valid. This signal indicates that a valid write response is available	70%
BREADY	I	Response ready. AXI master should ensure that this is always 1 if AXI-OCP bridge is connected to the DMA interface on the Scratchpad RAMs of a MIPS32 core	50%
Read Address Channel			
ARID['MBB_TAGID_WIDTH-1:0]	I	Read address ID. This signal is the identification tag for the Read address group of signals	30%
ARADDR['MBB_ADDR_WIDTH-1:0]	I	Read address. The Read address bus gives the address of the first transfer in a Read burst transaction	30%
ARLEN[3:0]	I	Burst Length. The burst length gives the exact number of transfers in a burst	30%
ARSIZE[2:0]	I	Burst size. This signal indicates the size of each transfer in the burst	30%
ARBURST[1:0]	I	Burst type	30%
ARLOCK[1:0]	I	Lock type. This signal provides additional information about the atomic characteristics of the transfer	Unused
ARCACHE[3:0]	I	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction	Unused

Table 8.5 AXI-OCP Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
ARPROT[2:0]	I	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access	Unused
ARVALID	I	Read address valid. This signal indicates that valid Read address and control information is available.	50%
ARSIDEBAND'[MBB_O2A_CONNIDSIDEBAND_WIDTH:0]	O	Sideband signal associated with the read command. The size of the signal is configurable by setting the value of 'MBB_SIDEBAND_WIDTH and 'MBB_O2A_CONNID_WIDTH defines in the configuration file (mbb_config.vh). The signal is not part of the AXI specification, and is used to pass user defined signals to the AXI domain (see *OC_MCmdSideBand*).	30%
ARREADY	O	Read address ready. This signal indicates that the slave is ready to accept and address and associated control information.	50%
Read Data Channel			
RID['MBB_TAGID_WIDTH-1:0]	O	Read ID tag. This signal is the ID tag of the read data group of signals	55%
*RDATA['MBB_BUS_DEFTYPE*64-1:0]*	O	Read data	55%
RRESP[1:0]	O	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR	55%
RLAST	O	Read last. This signal indicates the last transfer in a read burst	55%
RVALID	O	Read valid. This signal indicates that the required read data is available and the read transfer can complete	70%
RREADY	I	Read ready. This signal indicates that the master can accept the read data and response information. AXI master should ensure that this is always 1 if AXI-OCP bridge is connected to the DMA interface on the Scratchpad RAMs of a MIPS32 core.	50%
Miscellaneous			
SCANENABLE	I	Scan enable signal. Used to enable conditional registers during scan operation	30%
SCANMODE	I	Scan mode signal	30%
SCANIN[x:0]	I	Scan-in chains. The number of scan chains is determined by the value MBB_NUM_SCAN_CHAIN in the config file	30%
SCANOUT[x:0]	I	Scan-out chains. The number of scan chains is determined by the value MBB_NUM_SCAN_CHAIN in the config file	30%

Table 8.5 AXI-OCP Bridge Signals (Continued)

Signal Name	Dir	Description	Timing
'MBB_A2O_CFG_WRNP	SI	This static signal indicates the type of Write command that AXI-OCP puts on the OCP bus. The ScratchPad RAM's on MIPS cores support a WR (*OC_MCmd* = 3'b001) OCP command. The IO Coherence Unit on 1004K supports the WRNP (*OC_MCmd* = 3'b101) OCP command. Based on what is connected at the AXI-OCP's OCP master port, the port should be connected as follows: . ScratchPad RAM (ISPRAM/DSPRAM): 1'b0, a WR OCP command generated . 1004K's IO Coherence Unit (IOCU): 1'b1, WRNP OCP command generated	NA
OC_MRespAcceptEn	SI	This static signal indicates whether or not the attached OCP interface supports response flow control via *OC_MRespAccept*. The ScratchPad RAM's OCP DMA ports on MIPS cores do not support response flow control so this input must be set to 1'b0 for these interfaces. The slave DMA port of the IO Coherence Unit on 1004K does support response flow control so this input must be set to 1'b1 for this interface.	NA

Appendix A

References

This appendix lists documents that are referenced elsewhere in this document. They are available from MIPS.

1. MIPS® Physical Design Guide
   MIPS document: MD00606

2. MIPS® EJTAG Specification
   MIPS document: MD00047

3. MIPS32® 24K® and 24KE™ Processor Core Family Integrator's Guide
   MIPS document: MD00344

4. MIPS32® 34K® Processor Core Family Integrator's Guide
   MIPS document: MD00415

5. MIPS32® 74K™ Processor Core Family Integrator's Guide
   MIPS document: MD00499

6. MIPS32® 1004K™ CPU Family Integrator's Guide
   MIPS document: MD00620

7. MIPS32® 1004K™ Coherent Processing System User's Manual
   MIPS document: MD00597

8. MIPS32® P5600 Multiprocessing System Hardware User's Manual
   MIPS document: MD01026

9. MIPS® SOC-it® L2 Cache Controller Users Manual
   MIPS document: MD00525

10. MIPS® BusBridge™ 2 User's Manual
    MIPS document: MD00429

The following documents are available from Synopsys, Inc:

1. *Using the DesignWare Verification Models for the AMBA 3 AXI Protocol*, Version 5.20b, July 2008

2. *VMT User's Manual, Release 3.10a*, March 20, 2008

3. *DesignWare AMBA/AXI Verification IP Release Notes*, Release 5.20b, July 2008

4. *DesignWare Open Core Protocol (OCP) Verification IP HDL User Manual*, Version 1.50a, January 28, 2009

Appendix B

Revision History

MIPS documents include change bars (vertical bars in the page margin) that mark significant changes to the document since its last release. Change bars are removed for changes which are more than one revision old.

This document may refer to Architecture specifications (for example, instruction set descriptions and EJTAG register definitions), and change bars in these sections indicate changes since the previous version of the relevant Architecture document.

Revision	Date	Description
00.01	February 27, 2009	• Early access release
01.00	March 31, 2009	• General Availability
02.00	October 16, 2009	• Added the AXI-OCP bridge
02.01	March 7, 2011	• Changed document security classification from 1C (licensees only) to 2B (public domain).
02.02	April 17, 2014	• Increase address and data bus sizes. • Add support for P5600 core.